Amazon

(AWS Certified Security - Specialty SCS-C02)

AWS Certified Security - Specialty SCS-C02

Total: **295 Questions**

Link:

Question: 1

A company has an AWS Lambda function that creates image thumbnails from larger images. The Lambda function needs read and write access to an Amazon S3 bucket in the same AWS account.
Which solutions will provide the Lambda function this access? (Choose two.)

A.Create an IAM user that has only programmatic access. Create a new access key pair. Add environmental variables to the Lambda function with the access key ID and secret access key. Modify the Lambda function to use the environmental variables at run time during communication with Amazon S3.

B.Generate an Amazon EC2 key pair. Store the private key in AWS Secrets Manager. Modify the Lambda function to retrieve the private key from Secrets Manager and to use the private key during communication with Amazon S3.

C.Create an IAM role for the Lambda function. Attach an IAM policy that allows access to the S3 bucket.

D.Create an IAM role for the Lambda function. Attach a bucket policy to the S3 bucket to allow access. Specify the function's IAM role as the principal.

E.Create a security group. Attach the security group to the Lambda function. Attach a bucket policy that allows access to the S3 bucket through the security group ID.

Answer: CD

Explanation:

The correct answers are C and D.

Justification:

Option C: Create an IAM role for the Lambda function. Attach an IAM policy that allows access to the S3 bucket. This is the recommended and most secure approach for granting Lambda functions access to AWS resources. IAM roles provide temporary credentials to the Lambda function, eliminating the need to store long-term access keys. By attaching an IAM policy to the role, you can precisely define the permissions the Lambda function has on the S3 bucket (e.g., s3:GetObject, s3:PutObject). This follows the principle of least privilege.
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html

Option D: Create an IAM role for the Lambda function. Attach a bucket policy to the S3 bucket to allow access. Specify the function's IAM role as the principal. This is another valid approach. While attaching an IAM policy to the role itself (option C) is generally preferred for centralizing permissions management, bucket policies can also grant access to resources. Here, the bucket policy would explicitly allow the Lambda function's IAM role to perform actions on the bucket. Crucially, this bucket policy needs to specify the Lambda function's role's ARN as the principal. The effectiveness of this option relies on the role being properly configured, however, policies can be complex to debug.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-access-policy-language-overview.html

Why the other options are incorrect:

Option A: Create an IAM user that has only programmatic access. Create a new access key pair. Add environmental variables to the Lambda function with the access key ID and secret access key. Modify the Lambda function to use the environmental variables at run time during communication with Amazon S3. Storing access keys directly in environment variables of a Lambda function is highly discouraged. This is a security risk because access keys can be inadvertently exposed in logs or other application data. IAM roles offer a more secure, managed solution for granting temporary credentials.

Option B: Generate an Amazon EC2 key pair. Store the private key in AWS Secrets Manager. Modify the Lambda function to retrieve the private key from Secrets Manager and to use the private key during communication with Amazon S3. EC2 key pairs are for SSH access to EC2 instances, not for authenticating Lambda functions to S3. They are irrelevant in this context. Using them would introduce unnecessary complexity and security vulnerabilities.

Option E: Create a security group. Attach the security group to the Lambda function. Attach a bucket policy that allows access to the S3 bucket through the security group ID. Security groups are used to control network traffic, not to grant access to S3 buckets. Lambda functions, by default, don't reside within VPCs, so even if the function was in a VPC, security groups aren't the correct mechanism for authorizing access to S3.

Security groups authorize inbound/outbound traffic for compute instances or other resources. They do not directly govern access to S3 buckets.

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html

Question: 2

A security engineer is configuring a new website that is named example.com. The security engineer wants to secure communications with the website by requiring users to connect to example.com through HTTPS. Which of the following is a valid option for storing SSL/TLS certificates?

A.Custom SSL certificate that is stored in AWS Key Management Service (AWS KMS)

B.Default SSL certificate that is stored in Amazon CloudFront

C.Custom SSL certificate that is stored in AWS Certificate Manager (ACM)

D.Default SSL certificate that is stored in Amazon S3

Answer: C

Explanation:

The correct answer is C: Custom SSL certificate that is stored in AWS Certificate Manager (ACM). Here's why:

ACM is the preferred service for provisioning, managing, and deploying SSL/TLS certificates for use with AWS services and internally connected servers. It is designed to easily integrate with services like Elastic Load Balancing, CloudFront, API Gateway, and others, making it straightforward to secure websites and applications with HTTPS. Storing custom SSL/TLS certificates in ACM centralizes management, automates renewal, and helps ensure proper security practices.

Option A is incorrect because while AWS KMS can store cryptographic keys, it's not specifically designed for storing SSL/TLS certificates. ACM is the dedicated service for this purpose, offering features like automatic renewal and integration with other AWS services that KMS doesn't provide for SSL/TLS certificates.

Option B is incorrect because CloudFront can use ACM certificates, but it doesn't store default SSL certificates. Certificates for CloudFront are either imported to ACM or provisioned directly through ACM for use with CloudFront distributions. Default certificates might be used, but these are managed internally by CloudFront and not something the user directly interacts with or stores within CloudFront itself.

Option D is incorrect because Amazon S3 is an object storage service, not designed for storing and managing SSL/TLS certificates. While you could technically store a certificate file in S3, it would not provide the necessary management features like automatic renewal or integration with other AWS services. Furthermore, S3 is not intended to be a secure repository for private keys without significant additional configuration and safeguards. ACM is specifically built for secure certificate management.

Here are some resources for further reading:

AWS Certificate Manager (ACM):https://aws.amazon.com/certificate-manager/
ACM FAQs:https://aws.amazon.com/certificate-manager/faqs/
Using SSL/TLS Certificates with CloudFront:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/using-https-cloudfront-to-s3.html

Question: 3

A security engineer needs to develop a process to investigate and respond to potential security events on a company's Amazon EC2 instances. All the EC2 instances are backed by Amazon Elastic Block Store (Amazon EBS).
The company uses AWS Systems Manager to manage all the EC2 instances and has installed Systems Manager Agent (SSM Agent) on all the EC2 instances.
The process that the security engineer is developing must comply with AWS security best practices and must meet the following requirements:
A compromised EC2 instance's volatile memory and non-volatile memory must be preserved for forensic purposes.
A compromised EC2 instance's metadata must be updated with corresponding incident ticket information.
A compromised EC2 instance must remain online during the investigation but must be isolated to prevent the spread of malware.
Any investigative activity during the collection of volatile data must be captured as part of the process.
Which combination of steps should the security engineer take to meet these requirements with the LEAST operational overhead? (Choose three.)

A.Gather any relevant metadata for the compromised EC2 instance. Enable termination protection. Isolate the instance by updating the instance's security groups to restrict access. Detach the instance from any Auto Scaling groups that the instance is a member of. Deregister the instance from any Elastic Load Balancing (ELB) resources.

B.Gather any relevant metadata for the compromised EC2 instance. Enable termination protection. Move the instance to an isolation subnet that denies all source and destination traffic. Associate the instance with the subnet to restrict access. Detach the instance from any Auto Scaling groups that the instance is a member of. Deregister the instance from any Elastic Load Balancing (ELB) resources.

C.Use Systems Manager Run Command to invoke scripts that collect volatile data.

D.Establish a Linux SSH or Windows Remote Desktop Protocol (RDP) session to the compromised EC2 instance to invoke scripts that collect volatile data.

E.Create a snapshot of the compromised EC2 instance's EBS volume for follow-up investigations. Tag the instance with any relevant metadata and incident ticket information.

F.Create a Systems Manager State Manager association to generate an EBS volume snapshot of the compromised EC2 instance. Tag the instance with any relevant metadata and incident ticket information.

Answer: ACE

Explanation:

Here's a detailed justification for choosing options A, C, and E to address the security incident response scenario with the least operational overhead, while meeting all the requirements:

A - Metadata, Termination Protection, Isolation: Gathering metadata is crucial for context. Enabling termination protection prevents accidental deletion. Updating security groups to restrict access isolates the instance to contain the incident and prevent lateral movement. Removing the instance from Auto Scaling and ELB ensures it's no longer serving production traffic and won't be automatically replaced. This option directly addresses the requirement to isolate the instance.

C - Systems Manager Run Command for Volatile Data: Using Systems Manager Run Command is the most efficient and auditable way to collect volatile data (memory dumps, running processes, etc.). SSM Agent is already installed, making this method straightforward. It avoids the need to establish interactive SSH/RDP sessions, improving security and automation. Systems Manager also logs the commands executed, fulfilling the requirement to capture investigative activity. https://docs.aws.amazon.com/systems-manager/latest/userguide/execute-remote-commands.html

E - EBS Snapshot and Tagging: Creating an EBS snapshot preserves the non-volatile memory (disk contents) for later forensic analysis. Tagging the instance with metadata and incident ticket information ensures proper tracking and context. This directly addresses the preservation of non-volatile memory and the updating of metadata.

Why other options are not optimal:

B: Moving to an isolation subnet requires network configuration changes, increasing operational overhead compared to simply adjusting security groups.
D: Establishing SSH/RDP sessions is less secure and less auditable than using Systems Manager. It requires more manual intervention and increases the attack surface.
F: While State Manager can automate EBS snapshot creation, it adds complexity compared to a simple Run Command or manual snapshot. The question specifies least operational overhead.

In summary, ACE provide the most direct, automated, and auditable solution, leveraging existing Systems Manager infrastructure to collect volatile data, isolate the instance, and preserve non-volatile data, all while ensuring proper tracking through tagging. They follow AWS security best practices and minimize operational overhead.

Question: 4

A company has an organization in AWS Organizations. The company wants to use AWS CloudFormation StackSets in the organization to deploy various AWS design patterns into environments. These patterns consist of Amazon EC2 instances, Elastic Load Balancing (ELB) load balancers, Amazon RDS databases, and Amazon Elastic Kubernetes Service (Amazon EKS) clusters or Amazon Elastic Container Service (Amazon ECS) clusters. Currently, the company's developers can create their own CloudFormation stacks to increase the overall speed of delivery. A centralized CI/CD pipeline in a shared services AWS account deploys each CloudFormation stack.
The company's security team has already provided requirements for each service in accordance with internal standards. If there are any resources that do not comply with the internal standards, the security team must receive notification to take appropriate action. The security team must implement a notification solution that gives developers the ability to maintain the same overall delivery speed that they currently have.
Which solution will meet these requirements in the MOST operationally efficient way?

A.Create an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the security team's email addresses to the SNS topic. Create a custom AWS Lambda function that will run the aws cloudformation validate-template AWS CLI command on all CloudFormation templates before the build stage in the CI/CD pipeline. Configure the CI/CD pipeline to publish a notification to the SNS topic if any issues are found.

B.Create an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the security team's email addresses to the SNS topic. Create custom rules in CloudFormation Guard for each resource configuration. In the CI/CD pipeline, before the build stage, configure a Docker image to run the cfn-guard command on the CloudFormation template. Configure the CI/CD pipeline to publish a notification to the SNS topic if any issues are found.

C.Create an Amazon Simple Notification Service (Amazon SNS) topic and an Amazon Simple Queue Service (Amazon SQS) queue. Subscribe the security team's email addresses to the SNS topic. Create an Amazon S3 bucket in the shared services AWS account. Include an event notification to publish to the SQS queue when new objects are added to the S3 bucket. Require the developers to put their CloudFormation templates in the S3 bucket. Launch EC2 instances that automatically scale based on the SQS queue depth. Configure the EC2 instances to use CloudFormation Guard to scan the templates and deploy the templates if there are no issues. Configure the CI/CD pipeline to publish a notification to the SNS topic if any issues are found.

D.Create a centralized CloudFormation stack set that includes a standard set of resources that the developers can deploy in each AWS account. Configure each CloudFormation template to meet the security requirements.
For any new resources or configurations, update the CloudFormation template and send the template to the security team for review. When the review is completed, add the new CloudFormation stack to the repository for the developers to use.

Answer: B

Explanation:

The best solution is **B** because it provides the most operationally efficient way to validate CloudFormation templates against internal standards before deployment, while maintaining the developers' speed of delivery.

Here's why:

CloudFormation Guard:cfn-guard allows you to define rules (policies) to validate your CloudFormation templates. This allows for declarative validation, which is easier to maintain and update as standards evolve compared to custom scripts.

AWS CloudFormation Guard Documentation

Docker Image for Validation: Packaging cfn-guard and its dependencies into a Docker image ensures a consistent and reproducible validation environment across different CI/CD pipeline runners.

CI/CD Integration: Integrating the validation process directly into the CI/CD pipeline before the build stage ensures that only compliant templates are deployed, preventing non-compliant resources from being created.

SNS Notification: Using SNS to notify the security team provides a flexible and scalable way to alert them when non-compliant templates are detected.

Operational Efficiency: This approach avoids the operational overhead of managing EC2 instances for validation (option C) and the manual review process (option D). It also doesn't rely on potentially complex custom validation scripts (option A). CloudFormation Guard is purpose built for this scenario and therefore likely to require less custom scripting.

Here's why the other options are less suitable:

A: Using aws cloudformation validate-template only validates the syntax of the template, not the compliance with internal standards. Requires custom scripting and isn't as declarative as CloudFormation Guard.

C: Using SQS and EC2 instances adds unnecessary complexity and operational overhead. It also introduces potential scaling issues and latency. Developers have to put their templates in S3, which is unnecessary friction.

D: A centralized CloudFormation StackSet is a good idea for deploying standard resources, but it doesn't address the need to validate custom templates created by developers. This also severely hinders the speed of delivery.

Option B strikes the right balance between automation, compliance, and developer velocity. It leverages a purpose-built tool (cfn-guard) for validation, integrates seamlessly into the CI/CD pipeline, and provides timely notifications to the security team.

Question: 5

A company is migrating one of its legacy systems from an on-premises data center to AWS. The application server will run on AWS, but the database must remain in the on-premises data center for compliance reasons. The database is sensitive to network latency. Additionally, the data that travels between the on-premises data center and AWS must have IPsec encryption.
Which combination of AWS solutions will meet these requirements? (Choose two.)

A.AWS Site-to-Site VPN
B.AWS Direct Connect
C.AWS VPN CloudHub
D.VPC peering
E.NAT gateway

Answer: AB

Explanation:

The question requires establishing a secure, low-latency connection between an AWS application server and an on-premises database. The connection needs IPsec encryption.

Option A, AWS Site-to-Site VPN, is a correct choice because it creates an encrypted IPsec tunnel between the AWS environment and the on-premises data center. This satisfies the requirement for IPsec encryption. Site-to-Site VPN can be established over the internet or a dedicated connection, though the internet option may increase latency.

Option B, AWS Direct Connect, is also a correct choice. Direct Connect establishes a private, dedicated network connection from your on-premises data center to AWS. This dedicated connection bypasses the public internet, leading to lower latency and more consistent network performance, which is crucial for latency-sensitive database interactions. Direct Connect supports encrypting data in transit using MACsec, but it's important to configure IPsec for meeting the IPsec encryption requirement.

Option C, AWS VPN CloudHub, is incorrect because it's designed for connecting multiple remote sites using VPN connections, not for optimizing latency between a single on-premises data center and AWS.

Option D, VPC peering, is incorrect because it establishes a direct networking connection between two VPCs. It cannot connect an AWS VPC to an on-premises data center.

Option E, NAT gateway, is incorrect because it allows instances in a private subnet to connect to the internet or other AWS services, but it does not establish a secure, low-latency connection to an on-premises environment.

In summary, using AWS Direct Connect coupled with an IPsec VPN tunnel over Direct Connect offers the best combination of low latency and IPsec encryption for the required connection between AWS and the on-premises database. Site-to-Site VPN could be used alone, but may not provide the required low latency.

Relevant links:

AWS Site-to-Site VPN: https://aws.amazon.com/vpn/site-to-site-vpn/ AWS
Direct Connect: https://aws.amazon.com/directconnect/

Question: 6

A company has an application that uses dozens of Amazon DynamoDB tables to store data. Auditors find that the tables do not comply with the company's data protection policy.
The company's retention policy states that all data must be backed up twice each month: once at midnight on the 15th day of the month and again at midnight on the 25th day of the month. The company must retain the backups for 3 months.
Which combination of steps should a security engineer take to meet these requirements? (Choose two.)

A.Use the DynamoDB on-demand backup capability to create a backup plan. Configure a lifecycle policy to expire backups after 3 months.

B.Use AWS DataSync to create a backup plan. Add a backup rule that includes a retention period of 3 months.

C.Use AWS Backup to create a backup plan. Add a backup rule that includes a retention period of 3 months.

D.Set the backup frequency by using a cron schedule expression. Assign each DynamoDB table to the backup plan.

E.Set the backup frequency by using a rate schedule expression. Assign each DynamoDB table to the backup plan.

Answer: CD

Explanation:

The correct answer is CD. Here's why:

C. Use AWS Backup to create a backup plan. Add a backup rule that includes a retention period of 3

months. AWS Backup is the service specifically designed to centrally manage and automate backups across various AWS services, including DynamoDB. It allows defining backup plans with specific schedules and retention policies, fitting the company's needs precisely. By creating a backup rule with a 3-month retention period, the company ensures backups are stored for the required duration, meeting the retention policy.

D. Set the backup frequency by using a cron schedule expression. Assign each DynamoDB table to the backup plan. AWS Backup allows scheduling backups using cron expressions, providing precise control over the backup timing. A cron expression can easily define backups to occur at midnight on the 15th and 25th of each month, meeting the company's twice-monthly backup requirement. Assigning the DynamoDB tables to the backup plan ensures all tables are included in the scheduled backups.

Why the other options are incorrect:

A. Use the DynamoDB on-demand backup capability to create a backup plan. Configure a lifecycle policy to expire backups after 3 months. While DynamoDB offers on-demand backups and AWS Backup supports DynamoDB, the on-demand backup feature is not integrated to be used as a part of a lifecycle policy. You need a centralized backup solution like AWS Backup to create a comprehensive backup plan for multiple DynamoDB tables and manage the lifecycle policy.

B. Use AWS DataSync to create a backup plan. Add a backup rule that includes a retention period of 3 months. AWS DataSync is designed for moving large amounts of data between on-premises storage and AWS storage services or between AWS storage services. It is not primarily intended for backing up DynamoDB tables for compliance and retention purposes. AWS Backup is a better fit for these specific requirements.

E. Set the backup frequency by using a rate schedule expression. Assign each DynamoDB table to the backup plan. A rate schedule expression allows defining schedules in terms of fixed intervals (e.g., every 12 hours). However, it is less suitable for precise scheduling like the 15th and 25th of each month, making a cron expression a better choice for the given requirement.

Supporting Links:

AWS Backup:https://aws.amazon.com/backup/
AWS Backup - Scheduling Expressions:https://docs.aws.amazon.com/aws-backup/latest/devguide/schedule-expression.html
AWS DataSync:https://aws.amazon.com/datasync/
DynamoDB On-Demand Backup:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/backuprestore_HowItWorks.html

Question: 7

A company needs a security engineer to implement a scalable solution for multi-account authentication and authorization. The solution should not introduce additional user-managed architectural components. Native AWS features should be used as much as possible. The security engineer has set up AWS Organizations with all features activated and AWS IAM Identity Center (AWS Single Sign-On) enabled.
Which additional steps should the security engineer take to complete the task?

A.Use AD Connector to create users and groups for all employees that require access to AWS accounts. Assign AD Connector groups to AWS accounts and link to the IAM roles in accordance with the employees' job functions and access requirements. Instruct employees to access AWS accounts by using the AWS Directory Service user portal.

B.Use an IAM Identity Center default directory to create users and groups for all employees that require access to AWS accounts. Assign groups to AWS accounts and link to permission sets in accordance with the employees' job functions and access requirements. Instruct employees to access AWS accounts by using the IAM Identity Center user portal.

C.Use an IAM Identity Center default directory to create users and groups for all employees that require access to AWS accounts. Link IAM Identity Center groups to the IAM users present in all accounts to inherit existing permissions. Instruct employees to access AWS accounts by using the IAM Identity Center user portal.

D.Use AWS Directory Service for Microsoft Active Directory to create users and groups for all employees that require access to AWS accounts. Enable AWS Management Console access in the created directory and specify IAM Identity Center as a source of information for integrated accounts and permission sets. Instruct employees to access AWS accounts by using the AWS Directory Service user portal.

Answer: B

Explanation:

The correct answer is **B**. Here's a detailed justification:

The scenario requires a scalable, multi-account authentication and authorization solution using native AWS features, avoiding additional user-managed components. AWS IAM Identity Center (successor to AWS SSO) is already enabled within AWS Organizations.

Option B leverages IAM Identity Center's built-in directory for user and group management. This fulfills the "native AWS features" requirement and avoids the need for external directories or custom user management systems. The process involves creating users and groups directly within IAM Identity Center, aligning user access based on job functions. Crucially, assigning these groups to AWS accounts and linking them to permission sets defined within IAM Identity Center allows for centralized access control management. This aligns with the requirement for scalability. Employees then use the IAM Identity Center user portal for a streamlined single sign-on experience. This approach centralizes authentication and authorization. Permission sets are sets of AWS permissions defining the access level within linked AWS accounts, offering granular control and role-based access.

Option A is incorrect because while AD Connector could be used, it introduces more complexity than necessary and doesn't fully leverage the benefits of IAM Identity Center for single sign-on. AD Connector creates a directory gateway for routing requests to your existing on-premises Active Directory (AD) without replicating any of the AD data in the AWS Cloud. This is an architecture that is not needed here.

Option C is incorrect because linking IAM Identity Center groups to existing IAM users in each account undermines the centralized access control management that IAM Identity Center is designed to provide. It doesn't scale well and doesn't leverage permission sets for controlled access. Relying on pre-existing IAM users in accounts misses the centralized identity management IAM Identity Center offers.

Option D is incorrect because AWS Directory Service for Microsoft Active Directory, while capable of managing users and groups, introduces a separate managed Active Directory instance, which contradicts the requirement to minimize user-managed architectural components. It is an architectural design for a different use case. Linking IAM Identity Center as a "source of information" is not the primary approach for integrating user access. Instead, IAM Identity Center would be the IDP, not just a source of information for the created directory.

Therefore, option B presents the most suitable and efficient solution by leveraging IAM Identity Center's directory and permission sets for multi-account authentication and authorization.

Relevant links:

AWS IAM Identity Center: https://aws.amazon.com/iam/identity/
AWS Organizations: https://aws.amazon.com/organizations/
IAM Identity Center Permission Sets: https://docs.aws.amazon.com/singlesignon/latest/userguide/permission-sets.html

Question: 8

A company has deployed Amazon GuardDuty and now wants to implement automation for potential threats. The company has decided to start with RDP brute force attacks that come from Amazon EC2 instances in the company's AWS environment. A security engineer needs to implement a solution that blocks the detected communication from a suspicious instance until investigation and potential remediation can occur. Which solution will meet these requirements?

A.Configure GuardDuty to send the event to an Amazon Kinesis data stream. Process the event with an Amazon Kinesis Data Analytics for Apache Flink application that sends a notification to the company through Amazon Simple Notification Service (Amazon SNS). Add rules to the network ACL to block traffic to and from the suspicious instance.

B.Configure GuardDuty to send the event to Amazon EventBridge. Deploy an AWS WAF web ACL. Process the event with an AWS Lambda function that sends a notification to the company through Amazon Simple Notification Service (Amazon SNS) and adds a web ACL rule to block traffic to and from the suspicious instance.

C.Enable AWS Security Hub to ingest GuardDuty findings and send the event to Amazon EventBridge. Deploy AWS Network Firewall. Process the event with an AWS Lambda function that adds a rule to a Network Firewall firewall policy to block traffic to and from the suspicious instance.

D.Enable AWS Security Hub to ingest GuardDuty findings. Configure an Amazon Kinesis data stream as an event destination for Security Hub. Process the event with an AWS Lambda function that replaces the security group of the suspicious instance with a security group that does not allow any connections.

Answer: C

Explanation:

Here's a detailed justification for why option C is the most suitable solution for automating the response to RDP brute force attacks detected by GuardDuty, along with why the other options are less appropriate:

Why Option C is Correct:

GuardDuty and Security Hub Integration: Option C leverages the integration between GuardDuty and AWS Security Hub. GuardDuty detects threats, and Security Hub provides a centralized view and management of security findings across AWS services, making it a good starting point.

Event-Driven Architecture: Using Amazon EventBridge, the solution adopts an event-driven approach. EventBridge receives findings from Security Hub and triggers a Lambda function. This makes the response automated and scalable.

AWS Network Firewall for Blocking Traffic: The core of the solution involves using AWS Network Firewall, a fully managed network security service. The Lambda function dynamically adds a rule to the Network Firewall firewall policy to block traffic to and from the suspicious EC2 instance. Network Firewall operates at the network layer, providing centralized control and visibility over network traffic, making it ideal for blocking RDP brute force attempts.

Lambda Function for Automation: The Lambda function orchestrates the entire process. It receives the event from EventBridge, extracts the relevant information (e.g., the IP address of the attacking EC2 instance), and uses the AWS SDK to programmatically update the Network Firewall policy.

Why other options are incorrect:

Option A: Using Kinesis Data Streams and Kinesis Data Analytics is overkill for this scenario. While effective for processing large volumes of data, they add unnecessary complexity. Moreover, Network ACLs (Network Access Control Lists) are stateless, which makes implementing proper blocking logic complicated as it would require separate inbound and outbound rules. Using NACLs would not be able to block existing established sessions.

Option B: AWS WAF (Web Application Firewall) protects web applications (HTTP/HTTPS traffic) at Layer 7 of the OSI model. RDP brute force attacks occur over TCP on port 3389, which is not HTTP/HTTPS traffic. Therefore, WAF is not the appropriate tool for this type of attack.

Option D: Replacing the security group of the suspicious instance is a viable option to isolate the instance. However, it might cause disruption to other applications if they rely on the same security group. Also, it is not as surgical as using Network Firewall to only block malicious traffic.

Key Concepts and Links:

Amazon GuardDuty: Intelligent threat detection service. https://aws.amazon.com/guardduty/ **AWS Security Hub:** Centralized security management and compliance service.
https://aws.amazon.com/security-hub/
Amazon EventBridge: Serverless event bus for building event-driven applications.
https://aws.amazon.com/eventbridge/
AWS Lambda: Serverless compute service for running code without managing servers.
https://aws.amazon.com/lambda/
AWS Network Firewall: Managed network security service for protecting your virtual private clouds (VPCs).
https://aws.amazon.com/network-firewall/

Question: 9

A company has an AWS account that hosts a production application. The company receives an email notification that Amazon GuardDuty has detected an Impact:IAMUser/AnomalousBehavior finding in the account. A security engineer needs to run the investigation playbook for this security incident and must collect and analyze the information without affecting the application.
Which solution will meet these requirements MOST quickly?

A.Log in to the AWS account by using read-only credentials. Review the GuardDuty finding for details about the IAM credentials that were used. Use the IAM console to add a DenyAll policy to the IAM principal.

B.Log in to the AWS account by using read-only credentials. Review the GuardDuty finding to determine which API calls initiated the finding. Use Amazon Detective to review the API calls in context.

C.Log in to the AWS account by using administrator credentials. Review the GuardDuty finding for details about the IAM credentials that were used. Use the IAM console to add a DenyAll policy to the IAM principal.

D.Log in to the AWS account by using read-only credentials. Review the GuardDuty finding to determine which API calls initiated the finding. Use AWS CloudTrail Insights and AWS CloudTrail Lake to review the API calls in context.

Answer: B

Explanation:

The best solution is **B** because it provides a quick and non-disruptive method for investigating the GuardDuty finding.

Here's a detailed justification:

Read-only access: Logging in with read-only credentials (as suggested in options A, B, and D) is crucial for the investigation. This prevents accidental modifications to the production environment, minimizing the risk of disrupting the application.

GuardDuty finding details: The GuardDuty finding provides initial information about the IAM user and the anomalous activity detected. Reviewing these details is the first step in understanding the scope of the potential security incident.

Amazon Detective: Amazon Detective is specifically designed for security investigations. It automatically collects and analyzes log data (like CloudTrail logs) to provide a comprehensive view of security events. It allows security engineers to visualize relationships between users, roles, and resources, thus simplifying the investigation of the anomalous API calls identified in the GuardDuty finding. Detective helps to quickly understand the context surrounding the finding.

Option A's immediate DenyAll policy is premature: Immediately applying a DenyAll policy (as suggested in options A and C) is too drastic and can potentially disrupt legitimate application functionality. A thorough investigation should precede any restrictive actions.

Option C's administrator credentials are not necessary: Administrator credentials are not required to investigate GuardDuty findings and use Detective. Using read-only credentials adheres to the principle of least privilege.

Option D's CloudTrail Insights and Lake are less efficient: While CloudTrail Insights and Lake are powerful tools for analyzing CloudTrail data, they are not as purpose-built for security investigations as Amazon Detective. They may require more manual effort to correlate events and understand the context, making them less efficient in a situation where time is a factor. Detective provides an integrated and easier-to-use interface to examine the security-related logs.

In summary, option B combines the best practices of using read-only access for investigation with the efficiency of Amazon Detective to quickly understand the context of the anomalous API calls without impacting the production application.

Supporting Links:

Amazon GuardDuty:https://aws.amazon.com/guardduty/
Amazon Detective:https://aws.amazon.com/detective/
AWS CloudTrail:https://aws.amazon.com/cloudtrail/
IAM Best Practices:https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

Question: 10

Company A has an AWS account that is named Account A. Company A recently acquired Company B, which has an AWS account that is named Account B. Company B stores its files in an Amazon S3 bucket. The administrators need to give a user from Account A full access to the S3 bucket in Account B.
After the administrators adjust the IAM permissions for the user in Account A to access the S3 bucket in Account B, the user still cannot access any files in the S3 bucket.
Which solution will resolve this issue?

A.In Account B, create a bucket ACL to allow the user from Account A to access the S3 bucket in Account B.

B.In Account B, create an object ACL to allow the user from Account A to access all the objects in the S3 bucket in Account B.

C.In Account B, create a bucket policy to allow the user from Account A to access the S3 bucket in Account B.

D.In Account B, create a user policy to allow the user from Account A to access the S3 bucket in Account B.

Answer: C

Explanation:

The correct answer is C: In Account B, create a bucket policy to allow the user from Account A to access the S3 bucket in Account B. Here's why:

The problem describes a cross-account access scenario where a user in one AWS account (Account A) needs to access resources (an S3 bucket) in another AWS account (Account B). While IAM permissions in Account A allow the user to attempt to access the bucket, S3 buckets operate independently and have their own access control mechanisms. Therefore, even with proper IAM permissions in Account A, the user in Account A will be denied access to the S3 bucket in Account B unless explicit permission is granted within Account B.

Bucket policies are the primary mechanism for granting permissions to an S3 bucket itself. A bucket policy is a resource-based policy attached to the bucket in Account B. This policy explicitly allows the user from Account A to perform actions on the bucket. The policy would specify the Account A user's ARN and the allowed S3 actions (e.g., s3:GetObject, s3:PutObject).

Option A is incorrect because Bucket ACLs are less powerful and older access management methods. They are primarily for granting basic read/write access to AWS accounts, not to specific IAM users within an account. They also lack the granularity and condition options offered by bucket policies.

Option B is incorrect because Object ACLs require updating permissions for each individual object in the bucket, which is impractical and unmanageable, especially for buckets with many objects. While object ACLs can grant permissions, they don't scale well.

Option D is incorrect because creating a user policy within Account B is not the correct approach for cross-account access to an S3 bucket. Policies attach to IAM entities within that account (Account B), but are not designed to grant access to users that exist in other AWS accounts (Account A). The user's existing policy in Account A is already granting permission; the issue is the bucket in Account B has no policy allowing the user from Account A to access it.

Therefore, the most effective and scalable solution is to use a bucket policy within Account B that specifically allows the user from Account A to access the S3 bucket. This approach provides centralized and granular control over S3 bucket access.

For further research:

AWS S3 Bucket Policies:https://docs.aws.amazon.com/AmazonS3/latest/userguide/using-iam-policies.html **AWS IAM Policies:**https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html
Cross-Account Access:https://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_cross-account-with-roles.html

Question: 11

A company wants to receive an email notification about critical findings in AWS Security Hub. The company does not have an existing architecture that supports this functionality.
Which solution will meet the requirement?

A.Create an AWS Lambda function to identify critical Security Hub findings. Create an Amazon Simple Notification Service (Amazon SNS) topic as the target of the Lambda function. Subscribe an email endpoint to the SNS topic to receive published messages.

B.Create an Amazon Kinesis Data Firehose delivery stream. Integrate the delivery stream with Amazon EventBridge. Create an EventBridge rule that has a filter to detect critical Security Hub findings. Configure the delivery stream to send the findings to an email address.

C.Create an Amazon EventBridge rule to detect critical Security Hub findings. Create an Amazon Simple Notification Service (Amazon SNS) topic as the target of the EventBridge rule. Subscribe an email endpoint to the SNS topic to receive published messages.

D.Create an Amazon EventBridge rule to detect critical Security Hub findings. Create an Amazon Simple Email Service (Amazon SES) topic as the target of the EventBridge rule. Use the Amazon SES API to format the message. Choose an email address to be the recipient of the message.

Answer: C

Explanation:

The correct solution is **C: Create an Amazon EventBridge rule to detect critical Security Hub findings. Create an Amazon Simple Notification Service (Amazon SNS) topic as the target of the EventBridge rule. Subscribe an email endpoint to the SNS topic to receive published messages.**

Here's a detailed justification:

1. **EventBridge Integration:** AWS Security Hub integrates directly with Amazon EventBridge. Security Hub findings are automatically published as events to the EventBridge default event bus. This provides a near real-time mechanism for reacting to findings.
2. **Event Filtering:** EventBridge rules can be defined with specific filters to match events based on their content. In this scenario, the filter can be configured to specifically target Security Hub findings with

a severity level classified as "critical." This ensures that only important findings trigger a notification.

3. **SNS for Notification:** Amazon Simple Notification Service (SNS) is a fully managed messaging service. It's ideally suited for sending notifications to subscribers based on events.
4. **SNS Email Subscription:** SNS allows subscribers to receive notifications via various protocols, including email. By subscribing an email endpoint to the SNS topic, you ensure that whenever a message is published to the topic, an email is automatically sent to the designated email address. 5. **Simplicity and Scalability:** This approach is relatively simple to implement and maintain, leveraging managed services to reduce operational overhead. It's also highly scalable, as SNS can handle a large volume of notifications without requiring you to manage infrastructure.

Why other options are incorrect:

A (Lambda and SNS): While Lambda could be used to process Security Hub findings, it's not the most efficient or direct approach. EventBridge already provides the necessary filtering and routing capabilities, making Lambda redundant. Adding Lambda introduces complexity and maintenance overhead.

B (Kinesis Data Firehose and EventBridge): Kinesis Data Firehose is designed for streaming data to destinations like S3, Redshift, or Elasticsearch. It's not ideal for sending individual email notifications. While EventBridge could trigger Firehose, the delivery stream isn't designed for sending individual emails efficiently.

D (EventBridge and SES): While Amazon SES can send emails, using it directly as a target for EventBridge isn't the typical pattern. SNS provides a more robust and scalable notification mechanism, handling retries, fan-out scenarios, and different subscription types (not just email). Also, SES configuration may be more complex compared to SNS, especially managing sending authorization and reputation.

Authoritative Links:

Security Hub Integration with EventBridge:
https://docs.aws.amazon.com/securityhub/latest/userguide/securityhub-eventbridge.html
Amazon EventBridge:https://aws.amazon.com/eventbridge/
Amazon SNS:https://aws.amazon.com/sns/

Question: 12

An international company has established a new business entity in South Korea. The company also has established a new AWS account to contain the workload for the South Korean region. The company has set up the workload in the new account in the ap-northeast-2 Region. The workload consists of three Auto Scaling groups of Amazon EC2 instances. All workloads that operate in this Region must keep system logs and application logs for 7 years.
A security engineer must implement a solution to ensure that no logging data is lost for each instance during scaling activities. The solution also must keep the logs for only the required period of 7 years.
Which combination of steps should the security engineer take to meet these requirements? (Choose three.)

A.Ensure that the Amazon CloudWatch agent is installed on all the EC2 instances that the Auto Scaling groups launch. Generate a CloudWatch agent configuration file to forward the required logs to Amazon CloudWatch Logs.

B.Set the log retention for desired log groups to 7 years.

C.Attach an IAM role to the launch configuration or launch template that the Auto Scaling groups use. Configure the role to provide the necessary permissions to forward logs to Amazon CloudWatch Logs.

D.Attach an IAM role to the launch configuration or launch template that the Auto Scaling groups use. Configure the role to provide the necessary permissions to forward logs to Amazon S3.

E.Ensure that a log forwarding application is installed on all the EC2 instances that the Auto Scaling groups launch. Configure the log forwarding application to periodically bundle the logs and forward the logs to Amazon S3.

F.Configure an Amazon S3 Lifecycle policy on the target S3 bucket to expire objects after 7 years.

Answer: ABC

Explanation:

The correct answer is ABC because it provides a reliable and cost-effective solution for centralized logging with the required retention policy using native AWS services.

A. Ensuring the CloudWatch agent is installed and configured to forward logs: CloudWatch agent provides a streamlined way to collect logs from EC2 instances and send them to CloudWatch Logs. By installing the agent and configuring it to forward specific logs, you ensure that all logs from your EC2 instances, including those launched during scaling activities, are captured. This eliminates the need for custom log forwarding applications, simplifying the overall architecture.
[https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html]

B. Setting the log retention for desired log groups to 7 years: CloudWatch Logs allows you to define retention policies at the log group level. By configuring a 7-year retention policy for the log groups that receive the logs, you satisfy the requirement of keeping the logs for the specified duration and ensure compliance. This eliminates the need for separate log lifecycle management using services like S3.
[https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/RetentionPolicy.html]

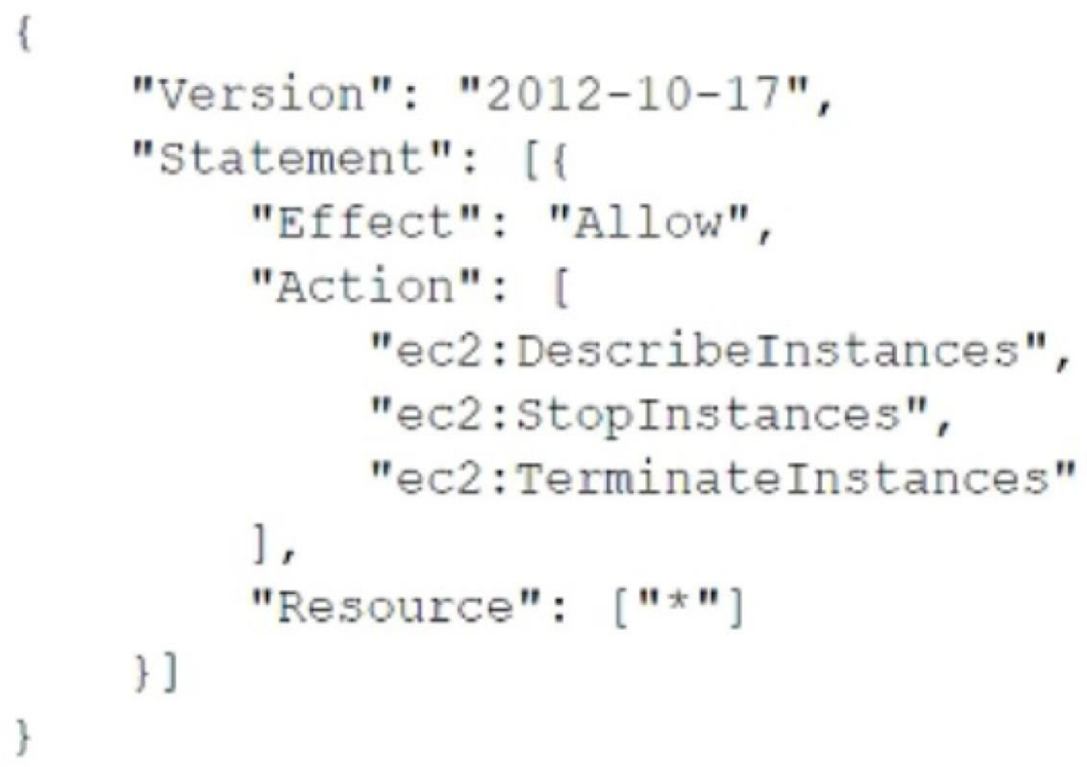
C. Attaching an IAM role with the necessary permissions: To enable the CloudWatch agent to send logs to CloudWatch Logs, you need to grant it the necessary IAM permissions. Attaching an IAM role to the launch configuration or launch template ensures that all EC2 instances launched by the Auto Scaling group automatically inherit the correct permissions to write to CloudWatch Logs. This ensures that instances can send logs immediately after startup, even during scaling events. This approach also follows the principle of least privilege.
[https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_ec2.html]

Why other options are incorrect:

D, E, and F: These options involve using S3 for log storage. While S3 is a valid option for long-term log storage, using CloudWatch Logs with built-in retention policies directly meets the requirements more efficiently and cost-effectively. Introducing S3 adds complexity and requires managing S3 lifecycle policies, which can be avoided with CloudWatch Logs. CloudWatch Logs provides near real-time log access.

Question: 13

A security engineer is designing an IAM policy to protect AWS API operations. The policy must enforce multi-factor authentication (MFA) for IAM users to access certain services in the AWS production account. Each session must remain valid for only 2 hours. The current version of the IAM policy is as follows:



```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "ec2:DescribeInstances",
            "ec2:StopInstances",
            "ec2:TerminateInstances"
        ],
        "Resource": ["*"]
    }]
}
```

Which combination of conditions must the security engineer add to the IAM policy to meet these requirements? (Choose two.)

A."Bool": "aws:MultiFactorAuthPresent": "true"
B."Bool": "aws:MultiFactorAuthPresent": "false"
C."NumericLessThan": "aws:MultiFactorAuthAge": "7200"
D."NumericGreaterThan": "aws:MultiFactorAuthAge": "7200"
E."NumericLessThan": "MaxSessionDuration": "7200"

Answer: AC

Explanation:

A. "Bool": "aws: Multi Factor Auth Present": "true"

C."NumericLessThan": "aws:MultiFactorAuthAge": "7200"

Question: 14

A company uses AWS Organizations and has production workloads across multiple AWS accounts. A security engineer needs to design a solution that will proactively monitor for suspicious behavior across all the accounts that contain production workloads.
The solution must automate remediation of incidents across the production accounts. The solution also must publish a notification to an Amazon Simple Notification Service (Amazon SNS) topic when a critical security finding is detected. In addition, the solution must send all security incident logs to a dedicated account.
Which solution will meet these requirements?

A.Activate Amazon GuardDuty in each production account. In a dedicated logging account, aggregate all GuardDuty logs from each production account. Remediate incidents by configuring GuardDuty to directly invoke an AWS Lambda function. Configure the Lambda function to also publish notifications to the SNS topic.

B.Activate AWS Security Hub in each production account. In a dedicated logging account, aggregate all Security Hub findings from each production account. Remediate incidents by using AWS Config and AWS Systems Manager. Configure Systems Manager to also publish notifications to the SNS topic.

C.Activate Amazon GuardDuty in each production account. In a dedicated logging account, aggregate all

GuardDuty logs from each production account. Remediate incidents by using Amazon EventBridge to invoke a custom AWS Lambda function from the GuardDuty findings. Configure the Lambda function to also publish notifications to the SNS topic.

D.Activate AWS Security Hub in each production account. In a dedicated logging account, aggregate all Security Hub findings from each production account. Remediate incidents by using Amazon EventBridge to invoke a custom AWS Lambda function from the Security Hub findings. Configure the Lambda function to also publish notifications to the SNS topic.

Answer: C

Explanation:

The correct answer is C because it provides a comprehensive and scalable solution for centralized security monitoring, automated remediation, and notification using GuardDuty and EventBridge. Here's a detailed justification:

GuardDuty for Threat Detection: GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect AWS accounts and workloads. Activating it in each production account ensures comprehensive threat detection across the entire environment. (https://aws.amazon.com/guardduty/)

Centralized Logging with GuardDuty: Aggregating GuardDuty logs into a dedicated logging account provides a centralized view of all security findings, enabling efficient analysis and investigation. This supports compliance requirements and simplifies incident response.

Automated Remediation with EventBridge and Lambda: EventBridge enables event-driven automation by routing GuardDuty findings to a custom Lambda function. The Lambda function can then execute remediation actions based on the specific finding, such as isolating compromised resources or blocking malicious IP addresses. This allows for automated incident response. (https://aws.amazon.com/eventbridge/)

Notification via Lambda and SNS: The Lambda function can also publish notifications to an SNS topic when a critical security finding is detected. This ensures that security personnel are promptly alerted to critical security events.

Why other options are incorrect:

Option A: While it uses GuardDuty, directly invoking Lambda from GuardDuty findings has limitations for complex remediation logic and customization. EventBridge provides a more flexible and robust event-driven architecture.

Option B: Security Hub aggregates security findings from multiple sources, including GuardDuty, but it primarily provides a unified view and compliance checks. While AWS Config and Systems Manager can be used for remediation, integrating directly with EventBridge offers more streamlined, event-driven automation.

Option D: Similar to Option B, Security Hub requires further configuration with EventBridge for streamlined, automated remediation. Although it also uses EventBridge and Lambda function, using GuardDuty as the primary source of threat detection aligns with the requirement of proactively monitoring for suspicious behavior. Security Hub aggregates findings from various sources and focuses more on compliance and consolidated security posture management.

In summary, Option C provides the most effective solution by combining the threat detection capabilities of GuardDuty, the event-driven automation of EventBridge, and the flexibility of Lambda functions for incident remediation and notification, all while centralizing security logs in a dedicated account.

Question: 15

A company is designing a multi-account structure for its development teams. The company is using AWS Organizations and AWS IAM Identity Center (AWS Single Sign-On). The company must implement a solution so that the development teams can use only specific AWS Regions and so that each AWS account allows access to only specific AWS services. Which solution will meet these requirements with the LEAST operational overhead?

A.Use IAM Identity Center to set up service-linked roles with IAM policy statements that include the Condition, Resource, and NotAction elements to allow access to only the Regions and services that are needed.

B.Deactivate AWS Security Token Service (AWS STS) in Regions that the developers are not allowed to use.

C.Create SCPs that include the Condition, Resource, and NotAction elements to allow access to only the Regions and services that are needed.

D.For each AWS account, create tailored identity-based policies for IAM Identity Center. Use statements that include the Condition, Resource, and NotAction elements to allow access to only the Regions and services that are needed.

Answer: C

Explanation:

The correct answer is C: Create SCPs that include the Condition, Resource, and NotAction elements to allow access to only the Regions and services that are needed. Here's why:

Service Control Policies (SCPs) offer centralized control over AWS accounts within an organization, allowing you to define guardrails that govern allowed actions. By attaching SCPs to the root, organizational units (OUs), or individual accounts, you can restrict the AWS Regions and services that users and roles within those accounts can access.

SCPs using the Condition, Resource, and NotAction elements provide granular control. The Condition element can restrict actions based on conditions like the AWS Region. The Resource element can limit actions to specific resources. The NotAction element denies specific actions while permitting all others. This centralized approach is ideal for enforcing consistent restrictions across multiple development accounts with minimal operational overhead.

Option A is less efficient because IAM Identity Center focuses on providing identity and access management. While you can configure permissions through IAM policies associated with IAM Identity Center, managing these policies at the IAM Identity Center level for region and service restrictions can become cumbersome and less scalable than SCPs, especially with many development teams and accounts. Service-linked roles are also not appropriate for managing access permissions for users.

Option B, deactivating AWS STS in specific Regions, is an incomplete solution. While it can prevent the creation of temporary security credentials in those Regions, it does not prevent users from using existing credentials or other access methods to potentially access services in those Regions. Also, disabling STS in certain regions may have unforeseen consequences for AWS services relying on it.

Option D involves creating tailored identity-based policies for IAM Identity Center for each account. This approach is highly repetitive and requires significant ongoing maintenance. Changes would need to be propagated to each account individually, increasing the risk of inconsistencies and operational overhead.

SCPs provide centralized governance, easier management, and reduce the administrative burden compared to configuring individual IAM policies or relying solely on AWS STS deactivation.Here are some links for more information:

AWS Organizations SCPs:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html **IAM Conditions:**
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition.html **AWS Identity and Access Management (IAM):**https://aws.amazon.com/iam/

Question: 16

A company is developing an ecommerce application. The application uses Amazon EC2 instances and an Amazon RDS MySQL database. For compliance reasons, data must be secured in transit and at rest. The company needs a solution that minimizes operational overhead and minimizes cost.
Which solution meets these requirements?

A.Use TLS certificates from AWS Certificate Manager (ACM) with an Application Load Balancer. Deploy self-signed certificates on the EC2 instances. Ensure that the database client software uses a TLS connection to Amazon RDS. Enable encryption of the RDS DB instance. Enable encryption on the Amazon Elastic Block Store (Amazon EBS) volumes that support the EC2 instances.

B.Use TLS certificates from a third-party vendor with an Application Load Balancer. Install the same certificates on the EC2 instances. Ensure that the database client software uses a TLS connection to Amazon RDS. Use AWS Secrets Manager for client-side encryption of application data.

C.Use AWS CloudHSM to generate TLS certificates for the EC2 instances. Install the TLS certificates on the EC2 instances. Ensure that the database client software uses a TLS connection to Amazon RDS. Use the encryption keys from CloudHSM for client-side encryption of application data.

D.Use Amazon CloudFront with AWS WAF. Send HTTP connections to the origin EC2 instances. Ensure that the database client software uses a TLS connection to Amazon RDS. Use AWS Key Management Service (AWS KMS) for client-side encryption of application data before the data is stored in the RDS database.

Answer: A

Explanation:

The correct answer is A. Here's a breakdown of why and why the other options are less suitable:

Why Option A is Correct:

TLS certificates from ACM with ALB: Using ACM provides free, managed TLS certificates for securing traffic between clients and the Application Load Balancer (ALB). This handles encryption in transit for web traffic, minimizing operational overhead related to certificate management. https://aws.amazon.com/certificate-manager/
Self-signed certificates on EC2 instances: Using self-signed certificates on EC2 instances to encrypt internal communications helps secure in transit traffic within the VPC between the ALB and your application instances. Self-signed certificates minimize cost when a full CA signed certificate is not needed.

Database client TLS connection: Enforcing a TLS connection from the EC2 instances to the RDS instance ensures that data is encrypted in transit between the application and the database. RDS supports TLS connections.
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html **RDS DB instance encryption:** Enabling encryption for the RDS DB instance ensures that the data is encrypted at rest within the database storage. This addresses the data at-rest compliance requirement.

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html
EBS volume encryption: Encrypting the EBS volumes for the EC2 instances ensures that data at rest on the underlying storage of the EC2 instances is protected.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html
Cost and Overhead: This solution effectively utilizes AWS managed services like ACM and RDS encryption, which are relatively inexpensive and minimize operational burden. Using self-signed certificates between the ALB and EC2 instances further reduces cost and overhead.

Why Other Options Are Incorrect:

Option B: Using a third-party certificate vendor adds cost and operational overhead compared to ACM. AWS Secrets Manager is not needed as the database already encrypts at rest.

Option C: CloudHSM is a dedicated hardware security module. Using it to generate TLS certificates is an overkill and significantly increases cost and complexity. Client-side encryption with CloudHSM keys introduces unnecessary complexity and isn't required if RDS encryption is enabled.

Option D: CloudFront is a CDN, more appropriate for caching static content, not as a core security component for an e-commerce application. Sending HTTP traffic to origin EC2 instances after passing through CloudFront/WAF does not satisfy the secure in transit requirement and is a vulnerability. Client-side encryption before storing data in RDS is not as efficient or manageable as enabling RDS native encryption.

In summary, option A provides a cost-effective and operationally efficient solution for securing data in transit and at rest by using AWS managed services where possible and minimizing unnecessary complexity.

Question: 17

A security engineer is working with a company to design an ecommerce application. The application will run on Amazon EC2 instances that run in an Auto Scaling group behind an Application Load Balancer (ALB). The application will use an Amazon RDS DB instance for its database.
The only required connectivity from the internet is for HTTP and HTTPS traffic to the application. The application must communicate with an external payment provider that allows traffic only from a preconfigured allow list of IP addresses. The company must ensure that communications with the external payment provider are not interrupted as the environment scales.
Which combination of actions should the security engineer recommend to meet these requirements? (Choose three.)

A.Deploy a NAT gateway in each private subnet for every Availability Zone that is in use.
B.Place the DB instance in a public subnet.
C.Place the DB instance in a private subnet.
D.Configure the Auto Scaling group to place the EC2 instances in a public subnet.
E.Configure the Auto Scaling group to place the EC2 instances in a private subnet.
F.Deploy the ALB in a private subnet.

Answer: ACE

Explanation:

Let's break down why the combination of A, C, and E is the correct solution for this AWS security scenario.

A. Deploy a NAT gateway in each private subnet for every Availability Zone that is in use. EC2 instances in the private subnet need to communicate with the external payment provider. Since this provider requires a preconfigured allow list of IP addresses, using a NAT Gateway is essential. A NAT Gateway allows instances in private subnets to initiate outbound internet traffic (like connecting to the payment provider) while preventing the internet from initiating connections to those instances. Deploying a NAT Gateway in each Availability Zone ensures high availability and avoids cross-AZ traffic costs. This fulfills the requirement of uninterrupted communication with the external payment provider during scaling.
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html

C. Place the DB instance in a private subnet. This is a critical security measure. The database should never be directly exposed to the internet. Placing it in a private subnet isolates it from external threats, only allowing access from resources within the VPC, such as the EC2 instances in the application tier.
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.Security.html

E. Configure the Auto Scaling group to place the EC2 instances in a private subnet. Placing the EC2 instances in a private subnet is a key security best practice. The EC2 instances running the application logic do not need direct internet access for incoming requests. The Application Load Balancer (ALB) handles the incoming HTTP/HTTPS traffic from the internet. The EC2 instances communicate with the ALB internally and use the NAT Gateway to access the external payment provider, thus maintaining isolation and security. This also works in tandem with NAT gateway usage.
https://docs.aws.amazon.com/autoscaling/ec2/userguide/asg-in-vpc.html

Let's examine why the other options are incorrect:

B. Place the DB instance in a public subnet. Incorrect. This is a significant security risk. Public subnets expose the database to the internet, making it vulnerable to attacks.

D. Configure the Auto Scaling group to place the EC2 instances in a public subnet. Incorrect. Placing EC2 instances directly in public subnets is generally discouraged for security reasons unless there is a compelling reason for them to have public IP addresses. The ALB already handles public traffic.

F. Deploy the ALB in a private subnet. Incorrect. An ALB needs to be in a public subnet to receive inbound HTTP/HTTPS traffic from the internet. It acts as the entry point for web traffic. While internal ALBs exist, they aren't appropriate for receiving traffic from the open internet.
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

Question: 18

A company uses several AWS CloudFormation stacks to handle the deployment of a suite of applications. The leader of the company's application development team notices that the stack deployments fail with permission errors when some team members try to deploy the stacks. However, other team members can deploy the stacks successfully.
The team members access the account by assuming a role that has a specific set of permissions that are necessary for the job responsibilities of the team members. All team members have permissions to perform operations on the stacks.
Which combination of steps will ensure consistent deployment of the stacks MOST securely? (Choose three.)

A.Create a service role that has a composite principal that contains each service that needs the necessary permissions. Configure the role to allow the sts:AssumeRole action.

B.Create a service role that has cloudformation.amazonaws.com as the service principal. Configure the role to allow the sts:AssumeRole action.

C.For each required set of permissions, add a separate policy to the role to allow those permissions. Add the ARN of each CloudFormation stack in the resource field of each policy.

D.For each required set of permissions, add a separate policy to the role to allow those permissions. Add the ARN of each service that needs the permissions in the resource field of the corresponding policy.

E.Update each stack to use the service role.
F Add a policy to each member role to allow the iam:PassRole action. Set the policy's resource field to the ARN of the service role.

Answer: BDE

Explanation:

Here's a detailed explanation of why options B, D, and E are the correct choices, and why the others are not, for ensuring consistent and secure CloudFormation stack deployments:

B. Create a service role that has cloudformation.amazonaws.com as the service principal. Configure the role to allow the sts:AssumeRole action.

This is a crucial step. CloudFormation needs a way to assume permissions to create and manage resources on your behalf. This service role grants CloudFormation the authority to perform actions defined within its policies. The principal cloudformation.amazonaws.com signifies that only the CloudFormation service is allowed to assume this role, limiting the blast radius and ensuring security. The sts:AssumeRole action is a prerequisite for CloudFormation to take on this role.https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-service.html

D. For each required set of permissions, add a separate policy to the role to allow those permissions. Add the ARN of each service that needs the permissions in the resource field of the corresponding policy.

This implements the principle of least privilege. The service role needs specific permissions to create the resources described in the CloudFormation template (e.g., EC2 instances, S3 buckets, IAM roles). Rather than granting broad access, you create specific policies that permit actions only on the necessary resources. The resource field should contain the ARNs of the services that will be affected by these actions within your templates, therefore the resource is a target service's ARN. This minimizes the risk of accidental or malicious actions.https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege

E. Update each stack to use the service role.

This explicitly tells CloudFormation to use the created service role when creating or updating the stack's resources. This is essential to ensure consistent deployment, regardless of who initiates the stack deployment. By using the service role, the stack uses a known set of permissions, rather than relying on the permissions of the individual user invoking the deployment. This mitigates the problem of some team members lacking the necessary permissions.https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-iam-service-role.html

Why the other options are incorrect:

A. Create a service role that has a composite principal that contains each service that needs the necessary permissions. Configure the role to allow the sts:AssumeRole action. This is overly complex and unnecessary. CloudFormation is the service deploying the resources, so it is the only principal that needs to assume the role directly. A composite principal is not the proper approach in this context.

C. For each required set of permissions, add a separate policy to the role to allow those permissions. Add the ARN of each CloudFormation stack in the resource field of each policy. This is incorrect. The resource field should contain the ARN of the resource being affected, not the CloudFormation stack itself. The stack is simply the tool managing the resources; the policies need to govern actions on the underlying AWS resources.

F. Add a policy to each member role to allow the iam:PassRole action. Set the policy's resource field to the ARN of the service role.iam:PassRole is only needed if you intend to allow users to pass the service role to other AWS services. In this case, CloudFormation itself assumes the role. You've already granted CloudFormation access to assume the service role and use it. The users do not need to be able to pass the service role to other AWS services themselves. So in this case, Option F is not required and may introduce unnecessary complications.

Question: 19

A company used a lift-and-shift approach to migrate from its on-premises data centers to the AWS Cloud. The company migrated on-premises VMs to Amazon EC2 instances. Now the company wants to replace some of components that are running on the EC2 instances with managed AWS services that provide similar functionality. Initially, the company will transition from load balancer software that runs on EC2 instances to AWS Elastic Load Balancers. A security engineer must ensure that after this transition, all the load balancer logs are centralized and searchable for auditing. The security engineer must also ensure that metrics are generated to show which ciphers are in use.
Which solution will meet these requirements?

A.Create an Amazon CloudWatch Logs log group. Configure the load balancers to send logs to the log group. Use the CloudWatch Logs console to search the logs. Create CloudWatch Logs filters on the logs for the required metrics.

B.Create an Amazon S3 bucket. Configure the load balancers to send logs to the S3 bucket. Use Amazon Athena to search the logs that are in the S3 bucket. Create Amazon CloudWatch filters on the S3 log files for the required metrics.

C.Create an Amazon S3 bucket. Configure the load balancers to send logs to the S3 bucket. Use Amazon Athena to search the logs that are in the S3 bucket. Create Athena queries for the required metrics. Publish the metrics to Amazon CloudWatch.

D.Create an Amazon CloudWatch Logs log group. Configure the load balancers to send logs to the log group. Use the AWS Management Console to search the logs. Create Amazon Athena queries for the required metrics. Publish the metrics to Amazon CloudWatch.

Answer: C

Explanation:

The correct solution is C. Here's why:

The core requirement is centralized, searchable load balancer logs with cipher usage metrics.

Centralized Logging and Search: AWS Elastic Load Balancers can directly send access logs to an Amazon S3 bucket. S3 provides durable storage for these logs. Amazon Athena can then be used to query these logs directly within S3 using SQL, making them searchable for auditing purposes. This addresses the "centralized and searchable logs" requirement.

Cipher Usage Metrics: Athena's SQL query capabilities can be leveraged to analyze the load balancer logs and extract information about the ciphers used during SSL/TLS connections. By crafting specific Athena queries, the security engineer can count the occurrences of each cipher and derive the required metrics. These metrics are not automatically generated by CloudWatch from the logs themselves.

Publishing Metrics: After extracting the cipher usage metrics using Athena, the results can be published to Amazon CloudWatch as custom metrics. This allows for visualization, alarming, and further analysis of the security posture.

Let's examine why the other options are less suitable:

A: While CloudWatch Logs can receive load balancer logs and be searched, it doesn't directly support extracting complex metrics like cipher usage from log data. CloudWatch Log Filters are suitable for simple pattern matching but not for generating aggregate metrics like cipher counts.

B: While storing logs in S3 and using Athena for search is correct, attempting to create CloudWatch filters on S3 files is not the intended use of CloudWatch. CloudWatch filters work on streams of log data, not directly on S3 files. CloudWatch Logs Insights could work here to extract metrics; however, it is more costly than Athena and, given the constraints of the question, less efficient.

D: CloudWatch Logs and Athena working together for the same logs would not be efficient and is not how logs are intended to flow between CloudWatch and Athena. CloudWatch does not have SQL-based querying, unlike Athena. Also, the AWS Management Console can search CloudWatch Logs, but that is not the strength provided by using Athena, which is complex SQL-based queries.

Therefore, option C provides the most efficient and direct way to achieve the stated requirements by leveraging the strengths of S3 for storage, Athena for querying and metric extraction, and CloudWatch for monitoring and alerting on the extracted metrics.

Relevant Documentation:

Elastic Load Balancing Access Logs:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-access-logs.html
Amazon Athena:https://aws.amazon.com/athena/
Amazon CloudWatch Custom Metrics:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/publishingMetrics.html

Question: 20

A company uses AWS Organizations to manage a multi-account AWS environment in a single AWS Region. The organization's management account is named management-01. The company has turned on AWS Config in all

accounts in the organization. The company has designated an account named security-01 as the delegated administrator for AWS Config.
All accounts report the compliance status of each account's rules to the AWS Config delegated administrator account by using an AWS Config aggregator. Each account administrator can configure and manage the account's own AWS Config rules to handle each account's unique compliance requirements.
A security engineer needs to implement a solution to automatically deploy a set of 10 AWS Config rules to all existing and future AWS accounts in the organization. The solution must turn on AWS Config automatically during account creation.
Which combination of steps will meet these requirements? (Choose two.)

A.Create an AWS CloudFormation template that contains the 10 required AWS Config rules. Deploy the template by using CloudFormation StackSets in the security-01 account.

B.Create a conformance pack that contains the 10 required AWS Config rules. Deploy the conformance pack from the security-01 account.

C.Create a conformance pack that contains the 10 required AWS Config rules. Deploy the conformance pack from the management-01 account.

D.Create an AWS CloudFormation template that will activate AWS Config. Deploy the template by using CloudFormation StackSets in the security-01 account.

E.Create an AWS CloudFormation template that will activate AWS Config. Deploy the template by using CloudFormation StackSets in the management-01 account.

Answer: BE

Explanation:

Here's a detailed justification for the answer choices B and E:

Choice B: Create a conformance pack that contains the 10 required AWS Config rules. Deploy the conformance pack from the security-01 account.

This is correct because AWS Config conformance packs provide a simplified way to package a collection of AWS Config rules and remediation actions that can be deployed as a single entity across an organization. Delegated administrators (like the security-01 account) can deploy conformance packs to member accounts.
This centralizes the management and deployment of Config rules across the organization. This method ensures all accounts, including existing ones, get the predefined set of AWS Config rules, promoting consistent compliance posture.

Choice E: Create an AWS CloudFormation template that will activate AWS Config. Deploy the template by using CloudFormation StackSets in the management-01 account.

This is correct because to automatically enable AWS Config in all existing and future accounts within an AWS Organization, you can leverage CloudFormation StackSets from the management account. StackSets allow you to deploy a CloudFormation template to multiple accounts and Regions with a single operation. In this case, the CloudFormation template would enable AWS Config. Because the template can be deployed to future accounts as part of the account creation process (using StackSets' automatic deployment features to new organizational units), it satisfies the requirement of automatic activation upon account creation. The management account must initiate this process to ensure organizational-wide setup.

Why other options are incorrect:

A: CloudFormation StackSets can deploy templates to multiple accounts, but using them directly to deploy individual config rules, especially when a solution like conformance packs is available, is less efficient and not the intended use case for the given scenario.
C: Although conformance packs can ensure consistency, they would be deployed from the delegated administrator account (security-01), according to the prompt, and not management-01.
D: While StackSets can deploy infrastructure, deploying the AWS Config activation template from the delegated administrator account doesn't follow the best practices of establishing foundational organizational policies from the management account. The management account should be responsible for the

organizational wide set up of AWS config.

Supporting links:

AWS Config Conformance Packs:https://docs.aws.amazon.com/config/latest/developerguide/conformance-packs.html
AWS CloudFormation StackSets:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/what-is-stacksets.html
AWS Organizations:https://aws.amazon.com/organizations/

Question: 21

A company has a legacy application that runs on a single Amazon EC2 instance. A security audit shows that the application has been using an IAM access key within its code to access an Amazon S3 bucket that is named DOC-EXAMPLE-BUCKET1 in the same AWS account. This access key pair has the s3:GetObject permission to all objects in only this S3 bucket. The company takes the application offline because the application is not compliant with the company's security policies for accessing other AWS resources from Amazon EC2.
A security engineer validates that AWS CloudTrail is turned on in all AWS Regions. CloudTrail is sending logs to an S3 bucket that is named DOC-EXAMPLE-BUCKET2. This S3 bucket is in the same AWS account as DOC-EXAMPLE-BUCKET1. However, CloudTrail has not been configured to send logs to Amazon CloudWatch Logs. The company wants to know if any objects in DOC-EXAMPLE-BUCKET1 were accessed with the IAM access key in the past 60 days. If any objects were accessed, the company wants to know if any of the objects that are text files (.txt extension) contained personally identifiable information (PII).
Which combination of steps should the security engineer take to gather this information? (Choose two.)

A.Use Amazon CloudWatch Logs Insights to identify any objects in DOC-EXAMPLE-BUCKET1 that contain PII and that were available to the access key.

B.Use Amazon OpenSearch Service to query the CloudTrail logs in DOC-EXAMPLE-BUCKET2 for API calls that used the access key to access an object that contained PII.

C.Use Amazon Athena to query the CloudTrail logs in DOC-EXAMPLE-BUCKET2 for any API calls that used the access key to access an object that contained PII.

D.Use AWS Identity and Access Management Access Analyzer to identify any API calls that used the access key to access objects that contained PII in DOC-EXAMPLE-BUCKET1.

E.Configure Amazon Macie to identify any objects in DOC-EXAMPLE-BUCKET1 that contain PII and that were available to the access key.

Answer: CE

Explanation:

Here's a detailed justification for the answer CE:

The question requires identifying if a specific IAM access key was used to access objects in an S3 bucket (DOC-EXAMPLE-BUCKET1) within the last 60 days and whether any accessed .txt files contained PII.

Option C is incorrect because it incorrectly assumes the CloudTrail logs can directly identify if an object contains PII. CloudTrail captures API calls, not the content of objects. Option D is incorrect because IAM Access Analyzer focuses on identifying unintended access to resources and doesn't analyze the content of accessed objects to detect PII. Option A is incorrect because CloudWatch Logs Insights works with CloudWatch Logs data, but CloudTrail is not configured to send logs to CloudWatch Logs, and CloudWatch Logs Insights does not have the ability to analyze the content of the object and find PII.

Option E, configuring Amazon Macie, is correct because Macie is a data security and data privacy service that uses machine learning and pattern matching to discover sensitive data, including PII, in S3 buckets. Macie can analyze the contents of objects in DOC-EXAMPLE-BUCKET1 to identify if any .txt files contain PII. It works independently of the access key used to access the objects, focusing solely on the data itself. This is specifically relevant as the company needs to know which objects contain PII, regardless of how they were accessed. https://aws.amazon.com/macie/

Option C, using Amazon Athena to query CloudTrail logs, is correct because CloudTrail logs contain records of API calls made to AWS services, including S3. By querying these logs, we can filter for API calls that used the specific IAM access key and accessed objects in DOC-EXAMPLE-BUCKET1 within the last 60 days. This confirms if the access key was indeed used to access objects in the specified bucket.

https://aws.amazon.com/athena/ and https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-concepts.html

Therefore, by combining the results of these two steps, the security engineer can determine if the access key was used to access objects in DOC-EXAMPLE-BUCKET1 and whether any accessed .txt files contain PII.

Question: 22

A security engineer creates an Amazon S3 bucket policy that denies access to all users. A few days later, the security engineer adds an additional statement to the bucket policy to allow read-only access to one other employee. Even after updating the policy, the employee sill receives an access denied message.
What is the likely cause of this access denial?

A.The ACL in the bucket needs to be updated.

B.The IAM policy does not allow the user to access the bucket.

C.It takes a few minutes for a bucket policy to take effect.

D.The allow permission is being overridden by the deny.

Answer: D

Explanation:

Here's a detailed justification for why option D is the most likely cause of the access denial, based on AWS security principles:

The core concept at play here is that, in AWS IAM and S3 bucket policies, explicit Deny statements always override Allow statements. This is a fundamental principle of how AWS authorization works. Even if an Allow statement grants a user permission to perform an action, if a Deny statement exists that applies to the same user and action, the Deny will take precedence.

In this scenario, the initial bucket policy explicitly denied all access to all users. Subsequently, an Allow statement was added to grant one employee read-only access. However, the original Deny statement is still in effect. Because it applies to all users, including the specified employee, the explicit Deny overrides the new Allow. Therefore, the employee continues to receive an access denied message.

Let's examine why the other options are less likely:

A. The ACL in the bucket needs to be updated: While Access Control Lists (ACLs) can control access to S3 buckets, bucket policies are generally the preferred method for managing S3 access permissions, offering more granular control. If the existing bucket policy is the source of the issue, manipulating ACLs is unlikely to solve it directly, and could add further complexity. Also, ACLs have limited capabilities.

B. The IAM policy does not allow the user to access the bucket: IAM policies apply to the user themselves. Bucket policies apply to the bucket. So, the user's IAM policy would generally allow for the user to perform certain actions. However, this problem exists because of a bucket policy, so it cannot be a problem with the IAM policy.

C. It takes a few minutes for a bucket policy to take effect: AWS services generally apply policy changes very quickly, with propagation usually occurring within seconds. A delay of days would be highly unusual and indicative of a more fundamental problem with the policy itself, rather than a simple propagation delay.

In summary, the most likely cause of the access denial is the existing Deny statement in the S3 bucket policy overriding the newly added Allow statement. To resolve this, the security engineer needs to modify the original Deny statement to exclude the employee from its scope, ensuring that the Allow statement can take effect.
This could involve adding a Condition block to the Deny statement that specifies the user should be exempt from the Deny, or removing the original Deny statement entirely, and rewriting the permissions.

Authoritative Links for further research:

AWS IAM Evaluation Logic:
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation_logic.html **S3 Bucket Policies:**https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html

Question: 23

A company is using Amazon Macie, AWS Firewall Manager, Amazon Inspector, and AWS Shield Advanced in its AWS account. The company wants to receive alerts if a DDoS attack occurs against the account.
Which solution will meet this requirement?

A.Use Macie to detect an active DDoS event. Create Amazon CloudWatch alarms that respond to Macie findings.

B.Use Amazon inspector to review resources and to invoke Amazon CloudWatch alarms for any resources that are vulnerable to DDoS attacks.

C.Create an Amazon CloudWatch alarm that monitors Firewall Manager metrics for an active DDoS event.

D.Create an Amazon CloudWatch alarm that monitors Shield Advanced metrics for an active DDoS event.

Answer: D

Explanation:

The correct answer is D because AWS Shield Advanced is specifically designed to protect AWS resources from DDoS attacks and provides detailed metrics related to such attacks. These metrics are readily available in Amazon CloudWatch.

Here's a detailed justification:

AWS Shield Advanced's Role: Shield Advanced offers enhanced DDoS protection compared to the standard AWS Shield. It provides detailed visibility into DDoS attacks targeting protected resources, including metrics and real-time event notifications.https://aws.amazon.com/shield/

CloudWatch Integration: Shield Advanced integrates seamlessly with Amazon CloudWatch, allowing you to monitor various metrics that indicate DDoS attack activity. These metrics include DDoSDetected, DDoSTotalRequestCount, and DDoSAttackVectors specific to resources protected by Shield Advanced.https://docs.aws.amazon.com/waf/latest/developerguide/ddos-metrics-cw.html

CloudWatch Alarms: By creating CloudWatch alarms that monitor these Shield Advanced metrics, you can be proactively alerted when a DDoS attack is detected. You can configure the alarm to trigger notifications via Amazon SNS, send alerts to security dashboards, or even automate mitigation actions.

Why other options are incorrect:

A (Macie): Amazon Macie focuses on discovering and protecting sensitive data, not detecting DDoS attacks. While it analyzes data access patterns, it's not designed to provide real-time DDoS alerts.

B (Inspector): Amazon Inspector assesses the security vulnerabilities of EC2 instances and container images. It identifies weaknesses that could be exploited in a DDoS attack, but it doesn't directly detect active attacks.

C (Firewall Manager): AWS Firewall Manager provides centralized management of AWS WAF rules and other security policies across multiple accounts. While it plays a role in mitigating DDoS attacks, it doesn't provide metrics specifically designed for DDoS detection like Shield Advanced does. Firewall Manager usually distributes the shield protections, but cloudwatch alarms for DDoS are configured on Shield metrics.

In summary, leveraging Shield Advanced metrics within CloudWatch alarms provides the most direct and effective method to receive alerts upon the occurrence of DDoS attacks against your AWS account.

Question: 24

A company hosts a web application on an Apache web server. The application runs on Amazon EC2 instances that are in an Auto Scaling group. The company configured the EC2 instances to send the Apache web server logs to an Amazon CloudWatch Logs group that the company has configured to expire after 1 year.
Recently, the company discovered in the Apache web server logs that a specific IP address is sending suspicious requests to the web application. A security engineer wants to analyze the past week of Apache web server logs to determine how many requests that the IP address sent and the corresponding URLs that the IP address requested. What should the security engineer do to meet these requirements with the LEAST effort?

A.Export the CloudWatch Logs group data to Amazon S3. Use Amazon Macie to query the logs for the specific IP address and the requested URL.

B.Configure a CloudWatch Logs subscription to stream the log group to an Amazon OpenSearch Service cluster. Use OpenSearch Service to analyze the logs for the specific IP address and the requested URLs.

C.Use CloudWatch Logs Insights and a custom query syntax to analyze the CloudWatch logs for the specific IP address and the requested URLs.

D.Export the CloudWatch Logs group data to Amazon S3. Use AWS Glue to crawl the S3 bucket for only the log entries that contain the specific IP address. Use AWS Glue to view the results.

Answer: C

Explanation:

The security engineer needs to analyze the last week's worth of Apache web server logs in CloudWatch to identify requests from a specific IP address and the URLs requested. The goal is to achieve this with the least amount of effort.

Option C, using CloudWatch Logs Insights, is the most efficient solution. CloudWatch Logs Insights allows direct querying and analysis of log data within CloudWatch Logs. This eliminates the need for exporting the data to other services. A custom query syntax can be used to filter the logs for the specific IP address and extract the corresponding URLs. This provides a quick and targeted analysis.

Option A involves exporting data to S3 and then using Macie. While Macie can perform data discovery and security risk analysis, it's primarily designed for sensitive data discovery and not for ad-hoc log analysis like this. Exporting to S3 and then using Macie adds unnecessary complexity and cost.

Option B suggests streaming logs to Amazon OpenSearch Service. While OpenSearch Service is excellent for log analytics, setting up a streaming subscription and configuring OpenSearch Service is an overkill for a one-time analysis of a week's worth of logs. This approach is more suitable for continuous monitoring and real-time analysis, not for a retrospective investigation.

Option D proposes exporting to S3 and using AWS Glue. Glue is a data catalog and ETL service, suitable for preparing and transforming data for analytics. It is not the most efficient tool for simply searching for specific entries within a log file. While Glue could potentially perform this task, it involves significantly more setup and configuration compared to CloudWatch Logs Insights. Glue is also unnecessary because we are not transforming or enriching the logs.

Therefore, CloudWatch Logs Insights provides the quickest and easiest way to analyze the logs directly, satisfying the "least effort" requirement.

Authoritative links:

CloudWatch Logs Insights:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html

Question: 25

While securing the connection between a company’s VPC and its on-premises data center, a security engineer sent a ping command from an on-premises host (IP address 203.0.113.12) to an Amazon EC2 instance (IP address 172.31.16.139). The ping command did not return a response. The flow log in the VPC showed the following:

```
2 123456789010 eni-1235b8ca 203.0.113.12 172.31.16.139 0 0 1 4 336 1432917027 1432917142 ACCEPT OK

2 123456789010 eni-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094 1432917142 REJECT OK
```

What action should be performed to allow the ping to work?

A.In the security group of the EC2 instance, allow inbound ICMP traffic.
B.In the security group of the EC2 instance, allow outbound ICMP traffic.
C.In the VPC’s NACL, allow inbound ICMP traffic.
D.In the VPC’s NACL, allow outbound ICMP traffic.

Answer: D

Explanation:

In the VPC’s NACL, allow outbound ICMP traffic.

Question: 26

A company developed an application by using AWS Lambda, Amazon S3, Amazon Simple Notification Service (Amazon SNS), and Amazon DynamoDB. An external application puts objects into the company's S3 bucket and tags the objects with date and time. A Lambda function periodically pulls data from the company's S3 bucket based on date and time tags and inserts specific values into a DynamoDB table for further processing. The data includes personally identifiable information (PII). The company must remove data that is older than 30 days from the S3 bucket and the DynamoDB table.
Which solution will meet this requirement with the MOST operational efficiency?

A.Update the Lambda function to add a TTL S3 flag to S3 objects. Create an S3 Lifecycle policy to expire objects that are older than 30 days by using the TTL S3 flag.

B.Create an S3 Lifecycle policy to expire objects that are older than 30 days. Update the Lambda function to add the TTL attribute in the DynamoDB table. Enable TTL on the DynamoDB table to expire entries that are older than 30 days based on the TTL attribute.

C.Create an S3 Lifecycle policy to expire objects that are older than 30 days and to add all prefixes to the S3 bucket. Update the Lambda function to delete entries that are older than 30 days.

D.Create an S3 Lifecycle policy to expire objects that are older than 30 days by using object tags. Update the Lambda function to delete entries that are older than 30 days.

Answer: B

Explanation:

Option B provides the most operationally efficient solution for automatically removing data older than 30 days from both the S3 bucket and the DynamoDB table.

Here's why:

S3 Lifecycle Policy: Creating an S3 Lifecycle policy allows for automated object expiration based on age. The policy can be configured to automatically delete objects older than 30 days. This eliminates the need for custom code to identify and delete these objects, thus reducing operational overhead.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lifecycle-management.html
DynamoDB TTL: DynamoDB Time To Live (TTL) enables automatic removal of expired items from a table. By adding a TTL attribute to the DynamoDB table (populated by the Lambda function when it inserts data), you can instruct DynamoDB to automatically delete entries that are older than 30 days based on the TTL attribute. This significantly simplifies data retention management and reduces operational complexity.
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html

The other options are less efficient:

Option A: Using a TTL S3 flag is not a standard S3 feature. While tags can be used in Lifecycle Policies, it's not a "TTL flag."
Option C: While an S3 lifecycle policy is used, the Lambda function is still responsible for deleting DynamoDB entries, which increases operational overhead and potential for errors. Adding all prefixes to the S3 bucket for lifecycle policy is not necessary if you can target based on object age.
Option D: Like option C, this requires Lambda to handle DynamoDB deletion, adding operational burden.

Therefore, option B leverages built-in AWS services to automate data expiration in both S3 and DynamoDB, resulting in the most operationally efficient solution and minimal custom code.

Question: 27

What are the MOST secure ways to protect the AWS account root user of a recently opened AWS account? (Choose two.)

A.Use the AWS account root user access keys instead of the AWS Management Console.

B.Enable multi-factor authentication for the AWS IAM users with the AdministratorAccess managed policy attached to them.

C.Use AWS KMS to encrypt all AWS account root user and AWS IAM access keys and set automatic rotation to 30 days.

D.Do not create access keys for the AWS account root user; instead, create AWS IAM users.

E.Enable multi-factor authentication for the AWS account root user.

Answer: DE

Explanation:

Here's a detailed justification for why options D and E are the most secure ways to protect the AWS account root user, along with supporting explanations and links:

The AWS account root user has unrestricted access to all resources in your AWS account. Its compromise represents a significant security risk. Therefore, securing it is paramount.

Option D: Do not create access keys for the AWS account root user; instead, create AWS IAM users.

Justification: Creating access keys for the root user provides a persistent and readily exploitable credential. If compromised, these keys grant full, unfettered access to the entire AWS environment. By avoiding the creation of root user access keys, you eliminate this specific attack vector. Instead, IAM users should be created and assigned specific permissions based on the principle of least privilege. This limits the impact of a compromised IAM user.

Concept: Least privilege is a security principle that dictates that users (or processes) should only have the minimum level of access needed to perform their job functions.

Reference:https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html **Why it's secure:** If you're not storing root user credentials anywhere (since you're not making keys), that removes a massive attack vector.

Option E: Enable multi-factor authentication for the AWS account root user.

Justification: Multi-factor authentication (MFA) adds an extra layer of security beyond a username and password. Even if the root user's password is compromised, an attacker would still need access to the second factor (e.g., a code from a mobile app or a security key) to gain access. This significantly reduces the risk of unauthorized access. AWS strongly recommends enabling MFA for the root user.

Concept: MFA requires users to provide multiple verification factors to prove their identity, bolstering protection against unauthorized access.

Reference:https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa.html
Why it's secure: MFA protects against password compromise, which is one of the most common security issues.

Why other options are incorrect:

A: Use the AWS account root user access keys instead of the AWS Management Console. This is explicitly bad advice. Never use root user access keys unless absolutely necessary. The AWS Management Console with MFA is the better path for the root user.

B: Enable multi-factor authentication for the AWS IAM users with the AdministratorAccess managed policy attached to them. While enabling MFA for IAM users is generally good, this is irrelevant to securing the root user, which is the focus of the question. Also, granting the AdministratorAccess policy should be avoided if possible.

C: Use AWS KMS to encrypt all AWS account root user and AWS IAM access keys and set automatic rotation to 30 days. Storing the root user access key anywhere is dangerous. Encrypting and rotating the root user access key is a complex operation and not recommended. Focus instead on eliminating the root user access key completely. While encrypting IAM access keys could be part of a comprehensive security strategy, it's not directly related to protecting the root user, and again, you should not store root credentials at all. It also doesn't address the fundamental problem of having the root access keys to begin with.

Question: 28

A company is expanding its group of stores. On the day that each new store opens, the company wants to launch a customized web application for that store. Each store's application will have a non-production environment and a production environment. Each environment will be deployed in a separate AWS account. The company uses AWS Organizations and has an OU that is used only for these accounts.
The company distributes most of the development work to third-party development teams. A security engineer needs to ensure that each team follows the company's deployment plan for AWS resources. The security engineer also must limit access to the deployment plan to only the developers who need access. The security engineer already has created an AWS CloudFormation template that implements the deployment plan.
What should the security engineer do next to meet the requirements in the MOST secure way?

A.Create an AWS Service Catalog portfolio in the organization's management account. Upload the CloudFormation template. Add the template to the portfolio's product list. Share the portfolio with the OU.

B.Use the CloudFormation CLI to create a module from the CloudFormation template. Register the module as a private extension in the CloudFormation registry. Publish the extension. In the OU, create an SCP that allows access to the extension.

C.Create an AWS Service Catalog portfolio in the organization's management account. Upload the

CloudFormation template. Add the template to the portfolio's product list. Create an IAM role that has a trust policy that allows cross-account access to the portfolio for users in the OU accounts. Attach the AWSServiceCatalogEndUserFullAccess managed policy to the role.

D.Use the CloudFormation CLI to create a module from the CloudFormation template. Register the module as a private extension in the CloudFormation registry. Publish the extension. Share the extension with the OU.

Answer: A

Explanation:

The correct answer is **A.** Here's why:

Requirement 1: Standardized Deployment Plan: The company needs to ensure all development teams adhere to a specific deployment plan defined by the CloudFormation template. AWS Service Catalog is specifically designed to provide centrally managed and governed catalogs of IT services that can be deployed across AWS accounts. By using Service Catalog, the security engineer can ensure consistency in deployments.

Requirement 2: Limited Access to Deployment Plan: The deployment plan (CloudFormation template) should be accessible only to authorized developers. Service Catalog allows you to control who has access to specific products (in this case, the CloudFormation template) through portfolio sharing and IAM permissions.

Why Option A is the Best Approach

1. **AWS Service Catalog Portfolio:** Creating a portfolio in the management account centralizes the CloudFormation template. This allows for easier management and versioning of the deployment plan.

2. **Uploading the CloudFormation template as a product:** This makes the deployment plan available as a self-service product that developers can launch.

3. **Sharing the portfolio with the OU:** Sharing the portfolio with the OU that contains the new store accounts makes the product available to all accounts within that OU, which aligns with the requirement that all store accounts have access to the deployment plan.

Why Other Options Are Less Suitable

Option B & D (CloudFormation Modules/Private Extension): While CloudFormation Modules allow you to create reusable CloudFormation code packages and Custom Resources allow you to extend CloudFormation's provisioning capabilities, they are more complex to manage for the given requirements than AWS Service Catalog. Distributing code and managing versions across different AWS accounts and development teams will become an administrative burden in the long run. The overhead is higher compared to leveraging Service Catalog. Also, Option B uses Service Control Policies (SCPs) to allow access, but SCPs are best used to set guardrails and prevent actions, rather than grant access.

Option C (IAM Role with Cross-Account Access): While creating an IAM role that allows users in the OU accounts to access the portfolio in the management account may seem like a reasonable choice, it is not ideal. The AWSServiceCatalogEndUserFullAccess policy grants a broader range of permissions than necessary. It is best to grant the least privilege, which sharing via Service Catalog inherently provides. Additionally, managing cross-account IAM roles across multiple accounts and developers increases the administrative burden.

Supporting Concepts and Links:

AWS Service Catalog: AWS Service Catalog allows organizations to create and manage catalogs of IT services that are approved for use on AWS. It enables you to centrally manage commonly deployed IT services and helps you achieve consistent governance and meet compliance requirements while enabling users to self-provision approved services.
https://aws.amazon.com/servicecatalog/
AWS Organizations: AWS Organizations helps you centrally manage and govern your environment as you grow and scale your AWS resources. https://aws.amazon.com/organizations/

CloudFormation Modules: CloudFormation Modules allow you to package and reuse parts of your CloudFormation templates. https://aws.amazon.com/blogs/aws/new-cloudformation-modules-build-reusable-templates-faster/

In summary, Option A provides the most secure and efficient way to meet the requirements by leveraging AWS Service Catalog to centrally manage the CloudFormation template, control access, and ensure consistent deployment across AWS accounts.

Question: 29

A team is using AWS Secrets Manager to store an application database password. Only a limited number of IAM principals within the account can have access to the secret. The principals who require access to the secret change frequently. A security engineer must create a solution that maximizes flexibility and scalability. Which solution will meet these requirements?

A.Use a role-based approach by creating an IAM role with an inline permissions policy that allows access to the secret. Update the IAM principals in the role trust policy as required.

B.Deploy a VPC endpoint for Secrets Manager. Create and attach an endpoint policy that specifies the IAM principals that are allowed to access the secret. Update the list of IAM principals as required.

C.Use a tag-based approach by attaching a resource policy to the secret. Apply tags to the secret and the IAM principals. Use the aws:PrincipalTag and aws:ResourceTag IAM condition keys to control access.

D.Use a deny-by-default approach by using IAM policies to deny access to the secret explicitly. Attach the policies to an IAM group. Add all IAM principals to the IAM group. Remove principals from the group when they need access. Add the principals to the group again when access is no longer allowed.

Answer: C

Explanation:

The correct answer is **C**. Here's a detailed justification:

Why Option C is Correct (Tag-Based Approach):

Flexibility and Scalability: Tags provide a highly flexible and scalable method for managing access control in AWS. Tags are key-value pairs that you can attach to AWS resources, including Secrets Manager secrets and IAM principals (users or roles).

Dynamic Access Control: The aws:PrincipalTag and aws:ResourceTag condition keys allow you to define fine-grained access control rules based on tags. As the IAM principals needing access to the secret change frequently, you can simply update the tags on the principals to grant or revoke access without modifying the secret's resource policy itself. This makes the solution very adaptable.

Reduced Policy Updates: This approach minimizes the need to modify the secret's resource policy. Instead of continuously updating the policy to add or remove individual IAM principals, you manage access through tag updates, simplifying administration and reducing the risk of errors.

Centralized Access Control: Resource policies associated with Secrets Manager secrets provide a centralized place to define access rules. By using tag-based conditions within the resource policy, you can enforce consistent access control across the organization.

Why Other Options are Incorrect:

Option A (Role-Based Approach with Inline Policy): While role-based access is generally a good practice, updating the role's trust policy whenever principals change is cumbersome and prone to errors. The trust policy identifies who can assume the role, and constantly modifying this policy would be operationally difficult.

Option B (VPC Endpoint Policy): VPC endpoint policies primarily control access to the Secrets Manager service through the VPC endpoint itself. They are not the appropriate mechanism for managing granular access to individual secrets based on IAM principals. They are better suited for controlling network access to the Secrets Manager service.

Option D (Deny-by-Default Approach): While a deny-by-default approach can be a good security practice, using IAM groups to manage access in this scenario is inefficient. Constantly adding and removing principals from the group would be operationally complex and increase the risk of accidental misconfiguration.

Authoritative Links for Further Research:

AWS Secrets Manager Resource Policies:
https://docs.aws.amazon.com/secretsmanager/latest/userguide/security_iam_id-based-policy-examples.html **IAM Condition Keys:**https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html
Tagging AWS Resources:https://docs.aws.amazon.com/general/latest/gr/aws_tagging.html

Question: 30

A company is hosting a web application on Amazon EC2 instances behind an Application Load Balancer (ALB). The application has become the target of a DoS attack. Application logging shows that requests are coming from a small number of client IP addresses, but the addresses change regularly.
The company needs to block the malicious traffic with a solution that requires the least amount of ongoing effort. Which solution meets these requirements?

A.Create an AWS WAF rate-based rule, and attach it to the ALB.
B.Update the security group that is attached to the ALB to block the attacking IP addresses.
C.Update the ALB subnet's network ACL to block the attacking client IP addresses.
D.Create an AWS WAF rate-based rule, and attach it to the security group of the EC2 instances.

Answer: A

Explanation:

The correct answer is A: Create an AWS WAF rate-based rule, and attach it to the ALB.

Here's a detailed justification:

AWS WAF (Web Application Firewall) is designed to protect web applications from common web exploits and bots that can affect availability, compromise security, or consume excessive resources. A rate-based rule in AWS WAF counts the requests coming from each IP address and blocks IP addresses that exceed a specified threshold within a defined period. This perfectly addresses the scenario where the attack originates from a changing set of IP addresses. By configuring a rate-based rule, the company can automatically block IPs that are sending an unusually high volume of requests, mitigating the DoS attack with minimal manual intervention. The ALB is the entry point for the web traffic, so attaching the WAF to it is the correct place.

Option B is incorrect because updating security groups manually with changing IP addresses is a reactive and labor-intensive approach that will not scale. Security groups are stateful firewalls operating at the instance level and aren't optimized for rapidly changing IP blocking scenarios.

Option C is incorrect because network ACLs (NACLs) are stateless firewalls that operate at the subnet level. While NACLs can block traffic, updating them frequently with changing IPs is cumbersome and error-prone. Moreover, NACLs are not as easily managed and do not offer the advanced features of WAF such as rate limiting.

Option D is incorrect because while it involves AWS WAF, attaching it directly to the EC2 instance security group bypasses the ALB, which is the intended entry point and the correct place for a web application firewall to analyze traffic and block malicious requests. The ALB is responsible for distributing traffic across instances, and a WAF at the ALB provides centralized protection.

In summary, AWS WAF's rate-based rule feature provides an automated and efficient way to block DoS attacks originating from a changing set of IP addresses with minimal ongoing effort. This aligns with the company's requirements and makes it the most suitable solution.

Here are some helpful links for further research:

AWS WAF:https://aws.amazon.com/waf/
AWS WAF Rate-Based Rules:https://docs.aws.amazon.com/waf/latest/developerguide/waf-rule-statement-type-rate-based.html
Application Load Balancer:https://aws.amazon.com/elasticloadbalancing/application-load-balancer/

Question: 31

A company has hundreds of AWS accounts in an organization in AWS Organizations. The company operates out of a single AWS Region. The company has a dedicated security tooling AWS account in the organization. The security tooling account is configured as the organization's delegated administrator for Amazon GuardDuty and AWS Security Hub. The company has configured the environment to automatically enable GuardDuty and Security Hub for existing AWS accounts and new AWS accounts.
The company is performing control tests on specific GuardDuty findings to make sure that the company's security team can detect and respond to security events. The security team launched an Amazon EC2 instance and attempted to run DNS requests against a test domain, example.com, to generate a DNS finding. However, the GuardDuty finding was never created in the Security Hub delegated administrator account.
Why was the finding was not created in the Security Hub delegated administrator account?

A.VPC flow logs were not turned on for the VPC where the EC2 instance was launched.

B.The VPC where the EC2 instance was launched had the DHCP option configured for a custom OpenDNS resolver.

C.The GuardDuty integration with Security Hub was never activated in the AWS account where the finding was generated.

D.Cross-Region aggregation in Security Hub was not configured.

Answer: B

Explanation:

The reason the GuardDuty finding wasn't created in the Security Hub delegated administrator account is likely due to the EC2 instance using a custom DNS resolver (option B). Here's why:

GuardDuty relies on VPC flow logs and DNS logs to detect malicious activity. When an EC2 instance performs a DNS lookup, GuardDuty analyzes these requests to identify potentially harmful domains. If the EC2 instance uses the default Amazon DNS server, GuardDuty has full visibility. However, if a custom DNS resolver, like OpenDNS, is configured via DHCP options in the VPC, the DNS requests are routed to that external resolver, bypassing GuardDuty's direct analysis of the DNS query itself.

GuardDuty primarily ingests DNS queries from the VPC's DNS resolver. By using a custom DNS resolver, the VPC's DNS resolution functionality isn't utilized, preventing GuardDuty from observing the DNS request to example.com and generating the expected finding. The query leaves the VPC before GuardDuty can analyze it.

While VPC Flow Logs (option A) are important for GuardDuty's functionality, DNS queries are primarily analyzed directly as opposed to relying on information gleaned solely from flow logs. The GuardDuty-Security Hub integration (option C) and Cross-Region aggregation (option D) are not directly related to GuardDuty's ability to detect DNS-related findings within a single AWS account and Region when the DNS resolution is external to AWS. They relate to propagation of the findings once they exist.

In summary, the EC2 instance bypassing the default Amazon DNS server prevented GuardDuty from seeing the DNS query for example.com and generating the anticipated finding.

Further research:

AWS GuardDuty DNS Log Monitoring:
https://docs.aws.amazon.com/guardduty/latest/ug/dns_log_monitoring.html
Amazon VPC DHCP Options Sets:
https://docs.aws.amazon.com/vpc/latest/userguide/VPC_DHCP_Options.html

Question: 32

An ecommerce company has a web application architecture that runs primarily on containers. The application containers are deployed on Amazon Elastic Container Service (Amazon ECS). The container images for the application are stored in Amazon Elastic Container Registry (Amazon ECR).
The company's security team is performing an audit of components of the application architecture. The security team identifies issues with some container images that are stored in the container repositories.
The security team wants to address these issues by implementing continual scanning and on-push scanning of the container images. The security team needs to implement a solution that makes any findings from these scans visible in a centralized dashboard. The security team plans to use the dashboard to view these findings along with other security-related findings that they intend to generate in the future. There are specific repositories that the security team needs to exclude from the scanning process.
Which solution will meet these requirements?

A.Use Amazon Inspector. Create inclusion rules in Amazon ECR to match repositories that need to be scanned. Push Amazon Inspector findings to AWS Security Hub.

B.Use ECR basic scanning of container images. Create inclusion rules in Amazon ECR to match repositories that need to be scanned. Push findings to AWS Security Hub.

C.Use ECR basic scanning of container images. Create inclusion rules in Amazon ECR to match repositories that need to be scanned. Push findings to Amazon Inspector.

D.Use Amazon Inspector. Create inclusion rules in Amazon Inspector to match repositories that need to be scanned. Push Amazon Inspector findings to AWS Config.

Answer: A

Explanation:

Here's a detailed justification for why option A is the correct answer, along with supporting concepts and links:

The requirement calls for continual and on-push scanning of container images in Amazon ECR, a centralized dashboard for findings, and the ability to exclude specific repositories from scanning.

Option A utilizes Amazon Inspector, which supports both on-push and continuous scanning of container images stored in ECR. Amazon Inspector assesses container images for software vulnerabilities and unintended network accessibility. The inclusion rules within Amazon ECR allows precise specification of which repositories should be included in the scan, effectively fulfilling the exclusion requirement. Amazon Inspector integrates seamlessly with AWS Security Hub. Security Hub aggregates security findings from various AWS services (including Inspector), providing a centralized dashboard to view security issues across the entire AWS environment. This fulfills the centralized dashboard requirement.

Option B suggests using ECR basic scanning, however, this option is not flexible enough to specify inclusion rules, which is a key requirement. ECR Basic scanning does not offer the ability to customize which repositories are scanned.

Option C also suggests ECR Basic scanning and pushes finding to Inspector, which doesn't satisfy the requirement of a centralized dashboard for all findings. Findings need to go to Security Hub for centralized reporting.

Option D proposes using Amazon Inspector but pushes findings to AWS Config. While AWS Config is a valuable service for compliance and configuration management, it is not designed to act as a centralized dashboard for security findings. Security Hub is the intended AWS service for this purpose.

Therefore, Option A is the only solution that correctly implements continual/on-push scanning with Amazon Inspector, allows for selective scanning through inclusion rules, and consolidates findings in AWS Security Hub for centralized visibility.

Supporting Links:

Amazon Inspector:https://aws.amazon.com/inspector/
AWS Security Hub:https://aws.amazon.com/security-hub/
Amazon ECR Image Scanning:https://docs.aws.amazon.com/AmazonECR/latest/userguide/image-scanning.html

Question: 33

A company has a single AWS account and uses an Amazon EC2 instance to test application code. The company recently discovered that the instance was compromised. The instance was serving up malware. The analysis of the instance showed that the instance was compromised 35 days ago.
A security engineer must implement a continuous monitoring solution that automatically notifies the company's security team about compromised instances through an email distribution list for high severity findings. The security engineer must implement the solution as soon as possible.
Which combination of steps should the security engineer take to meet these requirements? (Choose three.)

A.Enable AWS Security Hub in the AWS account.

B.Enable Amazon GuardDuty in the AWS account.

C.Create an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the security team's email distribution list to the topic.

D.Create an Amazon Simple Queue Service (Amazon SQS) queue. Subscribe the security team's email distribution list to the queue.

E.Create an Amazon EventBridge rule for GuardDuty findings of high severity. Configure the rule to publish a message to the topic.

F.Create an Amazon EventBridge rule for Security Hub findings of high severity. Configure the rule to publish a message to the queue.

Answer: BCE

Explanation:

Here's a detailed justification for why the combination of steps B, C, and E is the correct solution:

B. Enable Amazon GuardDuty in the AWS account: GuardDuty is a threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads. It analyzes VPC Flow Logs, CloudTrail event logs, and DNS logs to identify suspicious behavior, such as EC2 instances serving malware. Given the scenario states the EC2 instance was compromised and serving up malware, GuardDuty is perfectly suited to detect this kind of activity and alert the security team.
https://aws.amazon.com/guardduty/

C. Create an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the security team's email distribution list to the topic: SNS is a messaging service that allows you to send notifications to a variety of endpoints, including email. By creating an SNS topic and subscribing the security team's email distribution list, any message published to the topic will be sent to all members of the email list, facilitating rapid notification of security events. This satisfies the requirement for notifying the team.
https://aws.amazon.com/sns/

E. Create an Amazon EventBridge rule for GuardDuty findings of high severity. Configure the rule to publish a message to the topic: EventBridge (formerly CloudWatch Events) allows you to create rules that react to events in your AWS environment. GuardDuty findings are published as events. By creating an EventBridge rule that filters for high-severity GuardDuty findings and configures it to send a message to the SNS topic created in step C, you automate the process of notifying the security team only when a high-severity threat is detected. This ensures that the team is alerted to critical security incidents without unnecessary noise.

https://aws.amazon.com/eventbridge/

Why the other options are incorrect:

A. Enable AWS Security Hub in the AWS account: While Security Hub is a valuable security service, its primary function is to provide a central view of your security state across multiple AWS services and accounts and compliance with security best practices. It aggregates findings from other services like GuardDuty but doesn't directly detect the type of compromise described in the scenario as effectively and immediately as GuardDuty.

D. Create an Amazon Simple Queue Service (Amazon SQS) queue. Subscribe the security team's email distribution list to the queue: SQS is a queuing service, ideal for decoupling components of an application. While messages could be sent to a queue and then processed to send emails, it adds unnecessary complexity.

SNS provides a direct and more suitable mechanism for sending email notifications. Email subscriptions to SQS queues are not a native feature.

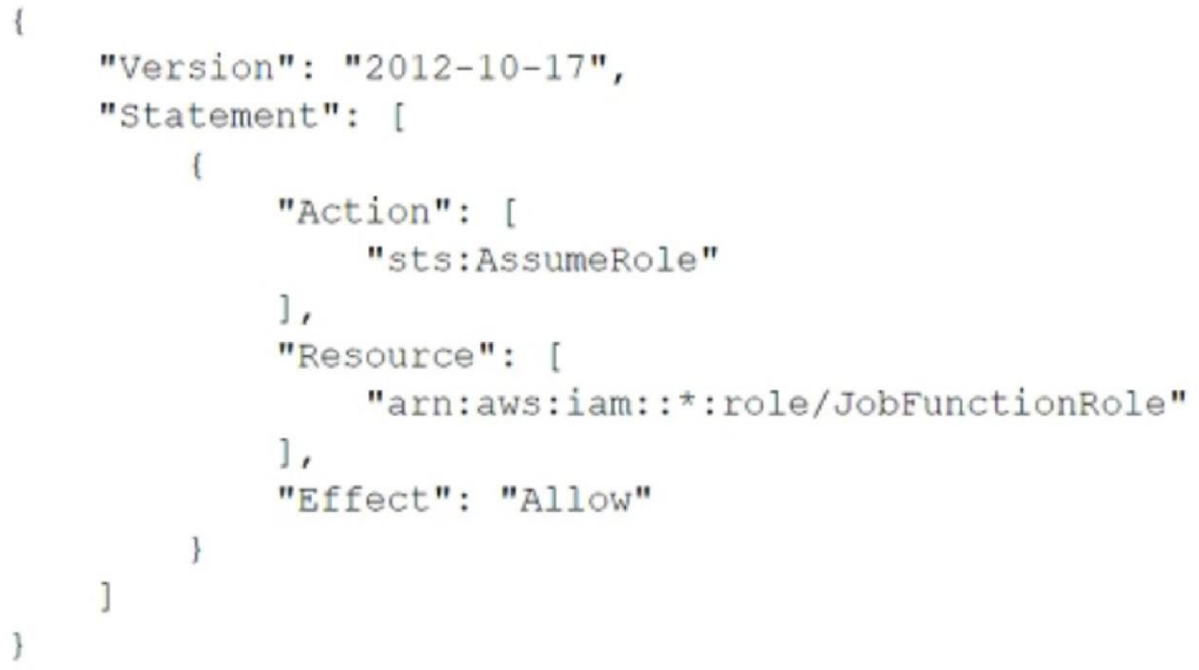
F. Create an Amazon EventBridge rule for Security Hub findings of high severity. Configure the rule to publish a message to the queue: While Security Hub can surface high severity findings, using it as the primary trigger adds a layer of indirection. Furthermore, using an SQS queue with a subsequent email mechanism is less efficient than directly utilizing SNS.

In summary, enabling GuardDuty for threat detection, using SNS for email notifications, and connecting them with EventBridge for automated alerting based on severity provides a rapid and effective solution to meet the requirements.

Question: 34

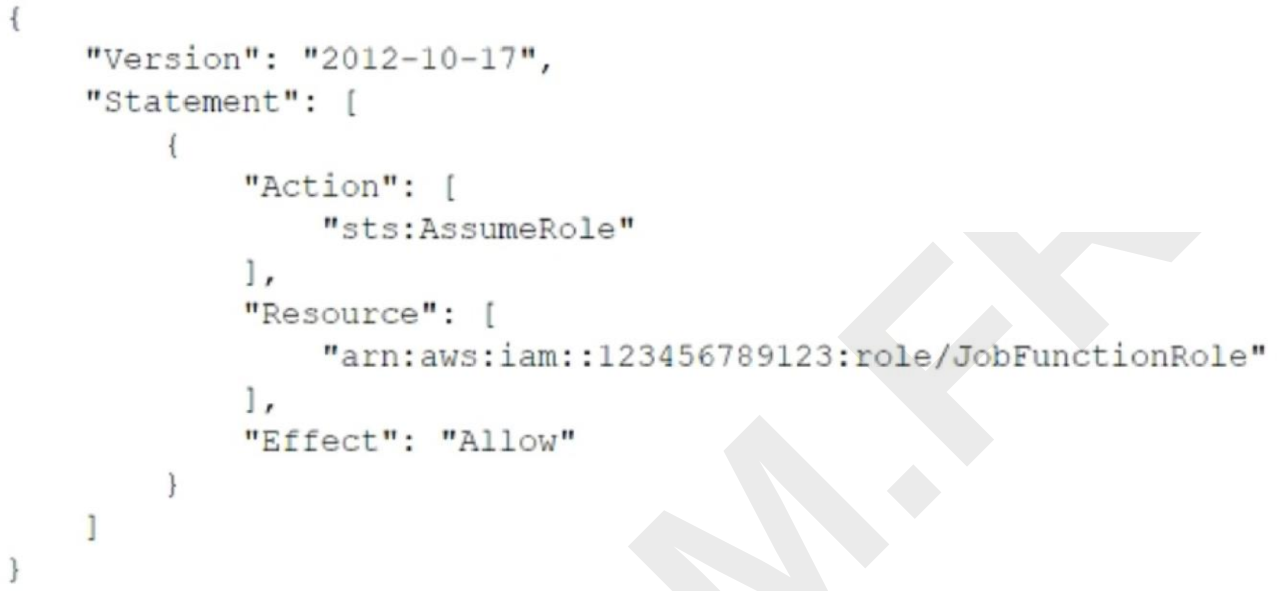
A company uses identity federation to authenticate users into an identity account (987654321987) where the users assume an IAM role named IdentityRole. The users then assume an IAM role named JobFunctionRole in the target AWS account (123456789123) to perform their job functions.

A user is unable to assume the IAM role in the target account. The policy attached to the role in the identity account is:

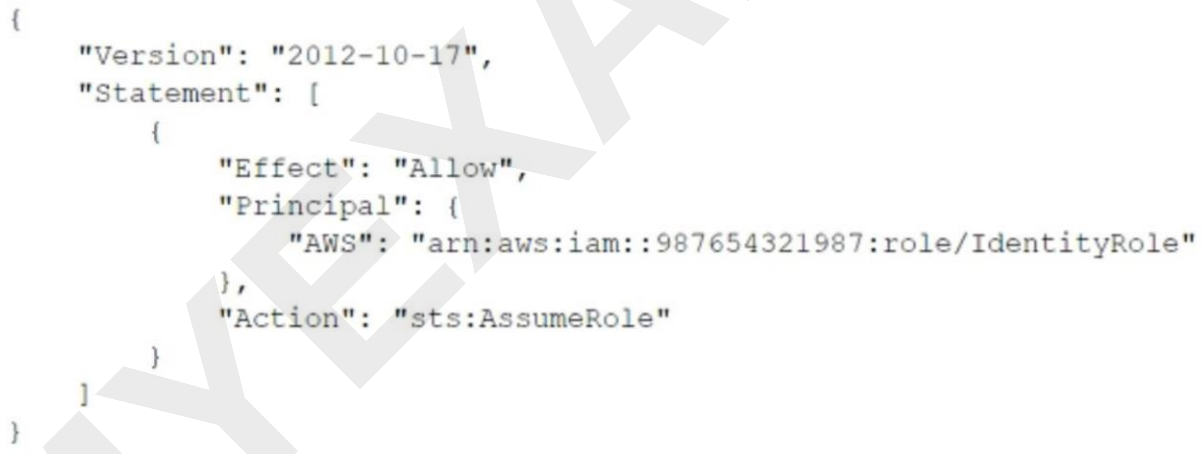

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/JobFunctionRole"
            ],
            "Effect": "Allow"
        }
    ]
}
```

What should be done to enable the user to assume the appropriate role in the target account?

A.Update the IAM policy attached to the role in the identity account to be:


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789123:role/JobFunctionRole"
            ],
            "Effect": "Allow"
        }
    ]
}
```

B.Update the trust policy on the role in the target account to be:


```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::987654321987:role/IdentityRole"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

C.Update the trust policy on the role in the identity account to be:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "AWS": "arn:aws:iam::987654321987:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

D.Update the IAM policy attached to the role in the target account to be:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1502946463000",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::123456789123:role/JobFunctionRole"
        }
    ]
}
```

Answer: B

Explanation:

Answer B

In IAM roles, use the Principal element in the role trust policy to specify who can assume the role.

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html

Question: 35

A company is using AWS Organizations to manage multiple AWS accounts for its human resources, finance, software development, and production departments. All the company's developers are part of the software development AWS account.
The company discovers that developers have launched Amazon EC2 instances that were preconfigured with software that the company has not approved for use. The company wants to implement a solution to ensure that developers can launch EC2 instances with only approved software applications and only in the software development AWS account.
Which solution will meet these requirements?

A.In the software development account, create AMIs of preconfigured instances that include only approved software. Include the AMI IDs in the condition section of an AWS CloudFormation template to launch the appropriate AMI based on the AWS Region. Provide the developers with the CloudFormation template to launch EC2 instances in the software development account.

B.Create an Amazon EventBridge rule that runs when any EC2 RunInstances API event occurs in the software development account. Specify AWS Systems Manager Run Command as a target of the rule. Configure Run Command to run a script that will install all approved software onto the instances that the developers launch.

C.Use an AWS Service Catalog portfolio that contains EC2 products with appropriate AMIs that include only approved software. Grant the developers permission to access only the Service Catalog portfolio to launch a product in the software development account.

D.In the management account, create AMIs of preconfigured instances that include only approved software. Use AWS CloudFormation StackSets to launch the AMIs across any AWS account in the organization. Grant the developers permission to launch the stack sets within the management account.

Answer: C

Explanation:

Here's a detailed justification for why option C is the best solution:

The problem requires controlling which software developers can deploy on EC2 instances within the software development AWS account. It also requires ensuring that the developers can only deploy EC2 instances with approved software.

Option C, using AWS Service Catalog, directly addresses these requirements. AWS Service Catalog allows organizations to create and manage catalogs of IT services that are approved for use on AWS. These services can include EC2 instances preconfigured with approved AMIs.

Here's why it's effective:

1. **Standardization:** Service Catalog promotes standardized deployments of approved resources. You can ensure that developers only deploy EC2 instances based on predefined, approved AMIs.
2. **Governance and Control:** Service Catalog enables central governance. By granting developers permission to access only the Service Catalog portfolio containing the approved EC2 products, you restrict their ability to launch instances with unapproved software. They can only launch what is defined in the Catalog.
3. **Self-Service Provisioning:** Developers can easily self-provision approved instances, simplifying their workflow while maintaining control.
4. **Auditing and Compliance:** Service Catalog helps maintain compliance by ensuring that only approved resources are deployed. You have a clear audit trail of what was deployed and by whom. 5. **Single AWS Account Scope:** Service Catalog focuses on the software development account, which aligns with the problem statement's requirement to control deployments only in that specific account.

Here's why the other options are less suitable:

Option A (CloudFormation Templates within software development account): While CloudFormation can be used, managing and updating templates to enforce approved AMIs can become complex. It doesn't provide the centralized governance and cataloging capabilities of Service Catalog, and it doesn't prevent developers from bypassing the template and launching instances directly.

Option B (EventBridge and Systems Manager Run Command): This approach reacts after the instance has been launched. While it can install approved software post-launch, it doesn't prevent the initial launch of an instance with unapproved software. Also, the target SSM Run Command can fail, resulting in the instance being deployed with unapproved software. This does not meet the requirement.

Option D (CloudFormation StackSets from the Management Account): This approach is overly complex. StackSets are generally used for deploying resources across multiple accounts, which is not a requirement in this scenario. Also, managing developers' permissions within the management account is generally undesirable, as it can introduce security risks. This will also provision EC2 instances across every single AWS account instead of just the software development account.

In summary, AWS Service Catalog offers the most comprehensive solution for controlling EC2 instance deployments within the software development AWS account, ensuring that developers can only use approved software while maintaining a streamlined self-service provisioning process.

Relevant Links:

AWS Service Catalog:https://aws.amazon.com/servicecatalog/
AWS Organizations:https://aws.amazon.com/organizations/

Question: 36

A company has enabled Amazon GuardDuty in all AWS Regions as part of its security monitoring strategy. In one of its VPCs, the company hosts an Amazon EC2 instance that works as an FTP server. A high number of clients from multiple locations contact the FTP server. GuardDuty identifies this activity as a brute force attack because of the high number of connections that happen every hour.
The company has flagged the finding as a false positive, but GuardDuty continues to raise the issue. A security engineer must improve the signal-to-noise ratio without compromising the company's visibility of potential anomalous behavior.
Which solution will meet these requirements?

A.Disable the FTP rule in GuardDuty in the Region where the FTP server is deployed.

B.Add the FTP server to a trusted IP list. Deploy the list to GuardDuty to stop receiving the notifications.

C.Create a suppression rule in GuardDuty to filter findings by automatically archiving new findings that match the specified criteria.

D.Create an AWS Lambda function that has the appropriate permissions to delete the finding whenever a new occurrence is reported.

Answer: C

Explanation:

The correct answer is C: Create a suppression rule in GuardDuty to filter findings by automatically archiving new findings that match the specified criteria.

Here's why:

GuardDuty is raising a false positive due to the legitimate high volume of connections to the FTP server. The goal is to reduce noise without losing visibility of actual threats. Suppression rules in GuardDuty provide a way to automatically archive findings that match specific criteria. This is ideal in situations where a legitimate activity triggers a finding that is not a real threat. By creating a rule based on characteristics of the FTP server connections (e.g., source IP ranges, destination port), the security engineer can automatically archive these findings, effectively suppressing the false positives.

Option A (disabling the FTP rule) is incorrect because it would eliminate the detection of any FTP-related attacks, not just the false positives. This reduces overall security visibility.

Option B (using a trusted IP list) is not suitable in this scenario. Trusted IP lists are intended for known, safe IP addresses and are not intended for dynamic or large sets of clients. The problem is not a malicious source IP, but the high volume of connections. Additionally, FTP clients may be coming from varying source IP addresses.

Option D (Lambda function to delete findings) is an unnecessary and inefficient solution. Deleting the finding programmatically doesn't address the root cause (the conditions triggering the false positive) and requires more complex configuration and maintenance than using built-in GuardDuty suppression rules. Suppression rules are the intended mechanism for addressing false positives in GuardDuty.

Suppression rules allow the security engineer to focus on genuine threats while acknowledging and automatically managing known exceptions. This approach balances security monitoring with practicality.

For more information on GuardDuty suppression rules:

AWS Documentation: https://docs.aws.amazon.com/guardduty/latest/ug/guardduty_findings_suppression-rules.html

Question: 37

A company is running internal microservices on Amazon Elastic Container Service (Amazon ECS) with the Amazon EC2 launch type. The company is using Amazon Elastic Container Registry (Amazon ECR) private repositories. A security engineer needs to encrypt the private repositories by using AWS Key Management Service (AWS KMS).
The security engineer also needs to analyze the container images for any common vulnerabilities and exposures (CVEs).
Which solution will meet these requirements?

A.Enable KMS encryption on the existing ECR repositories. Install Amazon Inspector Agent from the ECS container instances' user data. Run an assessment with the CVE rules.

B.Recreate the ECR repositories with KMS encryption and ECR scanning enabled. Analyze the scan report after the next push of images.

C.Recreate the ECR repositories with KMS encryption and ECR scanning enabled. Install AWS Systems Manager Agent on the ECS container instances. Run an inventory report.

D.Enable KMS encryption on the existing ECR repositories. Use AWS Trusted Advisor to check the ECS container instances and to verify the findings against a list of current CVEs.

Answer: B

Explanation:

The correct answer is B. Here's why:

Requirement 1: KMS Encryption of ECR Repositories

ECR repositories can be configured to use KMS for encrypting the image layers stored within them. This encryption is performed at rest, adding an extra layer of security to the container images. ECR uses KMS customer-managed keys (CMKs) or AWS-managed keys to encrypt the data. Enabling KMS encryption is critical for protecting sensitive data within the containers. Existing repositories cannot be directly modified to add KMS encryption; instead, they must be recreated.

Requirement 2: CVE Analysis of Container Images

ECR provides built-in image scanning capabilities that allow for the identification of common vulnerabilities and exposures (CVEs) in container images. When enabled, ECR automatically scans images after they are pushed to the repository. This scanning uses a vulnerability database to identify potential security flaws.

Why other options are incorrect:

A: You cannot enable KMS encryption on existing ECR repositories. They have to be created with the KMS encryption setting enabled. While Amazon Inspector can scan EC2 instances, it's not designed for direct container image scanning in ECR.

C: Systems Manager Agent and inventory reports are not the right tools for finding CVEs in container images. Systems Manager is more about instance management.

D: You cannot enable KMS encryption on existing ECR repositories. Trusted Advisor is a high-level service to analyze your AWS environment and find the ways to reduce cost, increase performance, or improve security. It cannot be used to find CVEs in container images.

Justification for option B:

Option B directly addresses both requirements efficiently:

1. **Recreation with KMS Encryption:** Creating new ECR repositories allows for enabling KMS encryption

during the creation process. This ensures that all subsequent images stored in the repository will be encrypted at rest using the specified KMS key.

2. **ECR Scanning Enabled:** Enabling ECR scanning on the newly created repositories ensures that every image pushed to the repository is automatically scanned for CVEs. The vulnerability scan results are available for review, allowing security engineers to identify and address potential security risks within the container images.

Relevant Links:

ECR Encryption:
ECR Image Scanning:

Question: 38

A company's security engineer has been tasked with restricting a contractor's IAM account access to the company's Amazon EC2 console without providing access to any other AWS services. The contractor's IAM account must not be able to gain access to any other AWS service, even if the IAM account is assigned additional permissions based on IAM group membership.
What should the security engineer do to meet these requirements?

A.Create an inline IAM user policy that allows for Amazon EC2 access for the contractor's IAM user.

B.Create an IAM permissions boundary policy that allows Amazon EC2 access. Associate the contractor's IAM account with the IAM permissions boundary policy.

C.Create an IAM group with an attached policy that allows for Amazon EC2 access. Associate the contractor's IAM account with the IAM group.

D.Create a IAM role that allows for EC2 and explicitly denies all other services. Instruct the contractor to always assume this role.

Answer: B

Explanation:

The correct answer is **B. Create an IAM permissions boundary policy that allows Amazon EC2 access. Associate the contractor's IAM account with the IAM permissions boundary policy.**

Here's why:

Permissions Boundaries: IAM permissions boundaries are advanced features used to limit the maximum permissions that an IAM entity (user or role) can have. A permissions boundary does not grant permissions; it limits them. Even if an IAM user or role is granted excessive permissions through policies attached directly or via groups, the permissions boundary acts as a constraint.

Meeting the Requirement: The problem states the contractor's IAM account must not be able to access any other AWS service, even if granted additional permissions via group membership. This is precisely what a permissions boundary achieves. By creating a boundary that only allows EC2 access, any attempts to grant additional permissions through IAM groups will be ineffective, as the boundary will restrict the effective permissions to only EC2.

Why other options are incorrect:

A. Inline IAM user policy: While an inline policy can grant EC2 access, it doesn't prevent the user from gaining additional permissions through other policies or group memberships, thus failing to meet the "no other AWS service" requirement.
C. IAM group with attached policy: Similar to option A, this grants EC2 access but doesn't prevent additional access from being granted through other group memberships or direct policies, failing the core constraint.

D. IAM role: While a role can restrict access, instructing the contractor to "always assume" the role doesn't guarantee they will. They could use their IAM user credentials directly, bypassing the role and its restrictions.

The requirement is to prevent access, not just advise against it. The problem also says the contractor's IAM account must have access, implying the contractor's IAM user credentials must be sufficient by themselves. Furthermore, a role is generally used for applications running on EC2 instances or other AWS services, not for human users, and doesn't directly prevent additional access from being granted via other policies.

Justification for Permissions Boundaries: Permissions boundaries offer a strong, centralized way to ensure an IAM entity never exceeds its intended permissions. This aligns perfectly with the requirement to limit the contractor's access solely to the EC2 console, regardless of other IAM configurations.

Therefore, using a permissions boundary is the only method that guarantees the contractor's IAM account will be restricted to EC2 access only, fulfilling the problem statement's requirement.

Relevant links for further research:

IAM Permissions Boundaries
Controlling Access to AWS Resources Using IAM

Question: 39

A company manages multiple AWS accounts using AWS Organizations. The company's security team notices that some member accounts are not sending AWS CloudTrail logs to a centralized Amazon S3 logging bucket. The security team wants to ensure there is at least one trail configured for all existing accounts and for any account that is created in the future. Which set of actions should the security team implement to accomplish this?

A.Create a new trail and configure it to send CloudTrail logs to Amazon S3. Use Amazon EventBridge to send notification if a trail is deleted or stopped.

B.Deploy an AWS Lambda function in every account to check if there is an existing trail and create a new trail, if needed.

C.Edit the existing trail in the Organizations management account and apply it to the organization.

D.Create an SCP to deny the cloudtrail:Delete* and cloudtrail:Stop* actions. Apply the SCP to all accounts.

Answer: C

Explanation:

The correct answer is **C. Edit the existing trail in the Organizations management account and apply it to the organization.**

Here's why this is the best approach, along with a detailed justification:

Justification:

1. **AWS Organizations and Organization Trails:** AWS Organizations allows centralized management of multiple AWS accounts. CloudTrail integrates seamlessly with Organizations. A key feature is the ability to create an organization trail within the management account (formerly known as the master account).
2. **Centralized Logging with Organization Trails:** An organization trail ensures that logs from all member accounts in the organization are captured and stored in a designated S3 bucket. This provides a single pane of glass for auditing and security analysis across the entire AWS environment.
3. **Automatic Application to New Accounts:** When you configure a trail to apply to the entire organization, CloudTrail automatically creates that trail in any new AWS accounts added to the

organization. This guarantees that all accounts, both present and future, are covered.

4. **Ease of Implementation:** Editing the existing trail in the management account is the simplest and most efficient way to achieve the desired outcome. It requires minimal code or scripting.

5. **Scalability and Maintainability:** This approach scales effectively as the organization grows because the trail is automatically deployed to new accounts. Maintenance is also simplified since you only need to manage a single trail in the management account.

6. **Control and Consistency:** An organization trail ensures that logging configuration is consistent across all accounts, reducing the risk of misconfiguration and improving overall security posture.

7. **Least Privilege Principle:** The organization trail is managed from the management account, adhering to the principle of least privilege by centralizing administrative control.

Why other options are not as suitable:

A: While creating a new trail and using EventBridge is partially helpful, it doesn't solve the core problem of automatically deploying the trail to all accounts, especially new ones.

B: Deploying a Lambda function in every account is an unnecessarily complex and difficult to maintain approach. It requires managing and updating the function in each account, increasing operational overhead.

D: While SCPs can prevent deletion/stopping of trails, they don't ensure a trail actually exists and is correctly configured. Moreover, simply denying deletion/stopping without ensuring the trail's existence can lead to unexpected logging gaps if the trail was never properly provisioned.

Authoritative Links:

Creating a Trail For an Organization:https://docs.aws.amazon.com/awscloudtrail/latest/userguide/creating-trail-organization.html
AWS Organizations:https://aws.amazon.com/organizations/

Question: 40

A company recently had a security audit in which the auditors identified multiple potential threats. These potential threats can cause usage pattern changes such as DNS access peak, abnormal instance traffic, abnormal network interface traffic, and unusual Amazon S3 API calls. The threats can come from different sources and can occur at any time. The company needs to implement a solution to continuously monitor its system and identify all these incoming threats in near-real time.
Which solution will meet these requirements?

A.Enable AWS CloudTrail logs, VPC flow logs, and DNS logs. Use Amazon CloudWatch Logs to manage these logs from a centralized account.

B.Enable AWS CloudTrail logs, VPC flow logs, and DNS logs. Use Amazon Macie to monitor these logs from a centralized account.

C.Enable Amazon GuardDuty from a centralized account. Use GuardDuty to manage AWS CloudTrail logs, VPC flow logs, and DNS logs.

D.Enable Amazon Inspector from a centralized account. Use Amazon Inspector to manage AWS CloudTrail logs, VPC flow logs, and DNS logs.

Answer: C

Explanation:

The correct answer is **C. Enable Amazon GuardDuty from a centralized account. Use GuardDuty to manage AWS CloudTrail logs, VPC flow logs, and DNS logs.**

Here's why:

The scenario describes a need for continuous monitoring and near real-time threat detection based on various log sources like CloudTrail, VPC Flow Logs, and DNS Logs. Amazon GuardDuty is specifically designed to fulfill this requirement. It's a threat detection service that continuously monitors your AWS accounts and workloads for malicious activity and unauthorized behavior.

GuardDuty analyzes these data sources:

AWS CloudTrail logs (management events): Detects unusual API calls and actions within your AWS environment.

VPC Flow Logs: Identifies suspicious network traffic patterns, such as unusual connection attempts or data transfers.

DNS logs: Detects malicious domain requests.

GuardDuty uses threat intelligence feeds, machine learning, and anomaly detection to identify threats. It automatically scales and requires no infrastructure management. A centralized account allows for consolidated monitoring across multiple AWS accounts, simplifying security administration.

Option A is incorrect because CloudWatch Logs is a log management service but doesn't provide automated threat detection capabilities based on anomaly detection and threat intelligence. It needs additional configuration and manual analysis.

Option B is incorrect because Amazon Macie is a data security and data privacy service that uses machine learning and pattern matching to discover and protect your sensitive data in AWS. It focuses on identifying and classifying sensitive data, not threat detection from logs and network traffic.

Option D is incorrect because Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. It focuses on finding vulnerabilities in applications, not on real-time threat detection based on log analysis.

Authoritative Links:

Amazon GuardDuty:https://aws.amazon.com/guardduty/
Amazon CloudTrail:https://aws.amazon.com/cloudtrail/
VPC Flow Logs:https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html
Amazon Macie:https://aws.amazon.com/macie/
Amazon Inspector:https://aws.amazon.com/inspector/

Question: 41

A company that uses AWS Organizations is using AWS IAM Identity Center (AWS Single Sign-On) to administer access to AWS accounts. A security engineer is creating a custom permission set in IAM Identity Center. The company will use the permission set across multiple accounts. An AWS managed policy and a customer managed policy are attached to the permission set. The security engineer has full administrative permissions and is operating in the management account. When the security engineer attempts to assign the permission set to an IAM Identity Center user who has access to multiple accounts, the assignment fails.
What should the security engineer do to resolve this failure?

A.Create the customer managed policy in every account where the permission set is assigned. Give the customer managed policy the same name and same permissions in each account.

B.Remove either the AWS managed policy or the customer managed policy from the permission set. Create a second permission set that includes the removed policy. Apply the permission sets separately to the user.

C.Evaluate the logic of the AWS managed policy and the customer managed policy. Resolve any policy conflicts in the permission set before deployment.

D.Do not add the new permission set to the user. Instead, edit the user's existing permission set to include the

AWS managed policy and the customer managed policy.

Answer: A

Explanation:

The correct answer is A. The failure in assigning the permission set to the IAM Identity Center user across multiple accounts arises because customer-managed policies are account-specific. When a permission set containing a customer-managed policy is used across multiple accounts within AWS Organizations, the policy must exist in each of those accounts. IAM Identity Center cannot automatically assume a policy from a different account. Therefore, the security engineer needs to replicate the customer-managed policy in every account where the permission set is intended to be used, ensuring it has the same name and identical permissions.

Option B is incorrect because creating multiple permission sets to workaround this limitation is not efficient and can lead to administrative overhead. Option C focuses on policy conflicts which aren't the cause of this assignment failure. While policy evaluation and conflict resolution are important for security best practices, they don't directly address the underlying problem of the customer-managed policy not existing in the target accounts. Option D is not a viable solution, as adding policies to the user's existing permission set doesn't change the fact that the customer-managed policy still needs to exist within each account where the permission set is applied. IAM Identity Center Permission sets need the custom managed policies deployed in each account.

Further reading on AWS IAM Identity Center and permission sets:

AWS documentation on Permission Sets:
https://docs.aws.amazon.com/singlesignon/latest/userguide/permission-sets.html
AWS documentation on Customer managed policies:
https://docs.aws.amazon.com/IAM/latest/UserGuide/access-policies-managed-vs-inline.html

Question: 42

A company has thousands of AWS Lambda functions. While reviewing the Lambda functions, a security engineer discovers that sensitive information is being stored in environment variables and is viewable as plaintext in the Lambda console. The values of the sensitive information are only a few characters long.
What is the MOST cost-effective way to address this security issue?

A.Set up IAM policies from the Lambda console to hide access to the environment variables.

B.Use AWS Step Functions to store the environment variables. Access the environment variables at runtime. Use IAM permissions to restrict access to the environment variables to only the Lambda functions that require access.

C.Store the environment variables in AWS Secrets Manager, and access them at runtime. Use IAM permissions to restrict access to the secrets to only the Lambda functions that require access.

D.Store the environment variables in AWS Systems Manager Parameter Store as secure string parameters, and access them at runtime. Use IAM permissions to restrict access to the parameters to only the Lambda functions that require access.

Answer: D

Explanation:

Here's a detailed justification for why option D is the most cost-effective solution for securely storing and accessing sensitive information within a large number of AWS Lambda functions:

The problem lies in storing sensitive information (environment variables) in plaintext within the Lambda console, exposing them to unauthorized access. To remediate this, the environment variables must be encrypted and access to them needs to be controlled. The requirement is to implement the solution in the most cost-effective way.

Option A is incorrect because IAM policies in the Lambda console can restrict who can view the environment variables, but they do not encrypt the values at rest. The variables are still stored as plaintext, so they would not resolve the security issue.

Option B is not ideal because AWS Step Functions is designed for orchestrating serverless workflows, not primarily for secret storage. Using Step Functions solely for environment variable storage would be an unnecessarily complex and expensive solution compared to alternatives.

Option C, using AWS Secrets Manager, is a valid solution. Secrets Manager is designed for securely storing secrets (API keys, passwords, etc.). However, it's generally more expensive than Parameter Store. Secrets Manager offers features like automatic rotation, which aren't necessary if the environment variables being stored don't require such functionality.

Option D, using AWS Systems Manager Parameter Store with secure strings, provides both encryption and access control at a lower cost than Secrets Manager, particularly for relatively static secrets. Parameter Store's secure string parameters are encrypted using AWS KMS, ensuring that sensitive information is not stored in plaintext. Access to these parameters can be finely controlled using IAM policies, allowing you to grant specific Lambda functions access only to the parameters they need. The cost is primarily associated with KMS usage, which is typically minimal for infrequent access. Because the question states that the values are only a few characters long, it becomes even more cost-effective to use Parameter Store, as Secrets Manager is really better suited to larger, regularly rotated secrets. The prompt's emphasis on "most cost-effective" points toward Parameter Store.

In summary, while both Secrets Manager and Parameter Store provide secure storage and access control, Parameter Store is the more cost-effective option in this specific scenario due to the small size and assumed static nature of the secrets. It addresses the security issue effectively by encrypting the data and using IAM for granular access control.

Relevant links:

AWS Systems Manager Parameter Store: https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html
AWS Secrets Manager: https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html AWS KMS: https://aws.amazon.com/kms/

Question: 43

A security engineer is using AWS Organizations and wants to optimize SCPs. The security engineer needs to ensure that the SCPs conform to best practices.
Which approach should the security engineer take to meet this requirement?

A.Use AWS IAM Access Analyzer to analyze the polices. View the findings from policy validation checks.
B.Review AWS Trusted Advisor checks for all accounts in the organization.
C.Set up AWS Audit Manager. Run an assessment for all AWS Regions for all accounts.
D.Ensure that Amazon Inspector agents are installed on all Amazon EC2 instances in all accounts.

Answer: A

Explanation:

The correct answer is A. Here's why:

IAM Access Analyzer is a service specifically designed to analyze IAM policies, including SCPs, and identify potential security issues. It leverages automated reasoning to validate policies against AWS best practices and generates findings on policy validation checks. These findings highlight deviations from best practices, such as overly permissive policies or policies that grant unintended access. By analyzing the SCPs with IAM Access Analyzer, the security engineer can proactively identify and remediate any security gaps, ensuring the SCPs conform to AWS's recommended guidelines. This provides a direct assessment of policy effectiveness.

Option B, reviewing AWS Trusted Advisor, offers high-level recommendations across cost optimization, security, fault tolerance, and performance, but its checks are broader and less granular than the policy-specific analysis provided by IAM Access Analyzer. While Trusted Advisor includes some security checks, it doesn't delve into the specific logic of individual SCPs.

Option C, setting up AWS Audit Manager, focuses on auditing compliance against regulatory standards and industry frameworks. While Audit Manager is valuable for compliance, it does not directly provide feedback on optimizing SCPs or ensuring they conform to best practices related to access control.

Option D, ensuring Amazon Inspector agents are installed, is relevant for identifying vulnerabilities in EC2 instances, but does not address SCP optimization. Inspector assesses the security posture of your EC2 instances, but not the organizational-level access control defined by SCPs.

Therefore, using IAM Access Analyzer provides the most direct and effective approach to ensure SCPs conform to best practices by providing policy-level findings.

Relevant links:

AWS IAM Access Analyzer: https://aws.amazon.com/iam/features/access-analyzer/
AWS Organizations SCPs:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html

Question: 44

A company uses Amazon RDS for MySQL as a database engine for its applications. A recent security audit revealed an RDS instance that is not compliant with company policy for encrypting data at rest. A security engineer at the company needs to ensure that all existing RDS databases are encrypted using server-side encryption and that any future deviations from the policy are detected.
Which combination of steps should the security engineer take to accomplish this? (Choose two.)

A.Create an AWS Config rule to detect the creation of unencrypted RDS databases. Create an Amazon EventBridge rule to trigger on the AWS Config rules compliance state change and use Amazon Simple Notification Service (Amazon SNS) to notify the security operations team.

B.Use AWS System Manager State Manager to detect RDS database encryption configuration drift. Create an Amazon EventBridge rule to track state changes and use Amazon Simple Notification Service (Amazon SNS) to notify the security operations team.

C.Create a read replica for the existing unencrypted RDS database and enable replica encryption in the process. Once the replica becomes active, promote it into a standalone database instance and terminate the unencrypted database instance.

D.Take a snapshot of the unencrypted RDS database. Copy the snapshot and enable snapshot encryption in the process. Restore the database instance from the newly created encrypted snapshot. Terminate the unencrypted database instance.

E.Enable encryption for the identified unencrypted RDS instance by changing the configurations of the existing database.

Answer: AD

Explanation:

Let's analyze why options A and D are the correct choices.

Option A: AWS Config and EventBridge for Detection and Notification

This option effectively addresses the requirement for detecting deviations from the encryption policy in both existing and future RDS instances. AWS Config can be configured with a rule to continuously assess RDS instances and determine whether they are encrypted. If an unencrypted instance is detected, the rule will mark it as non-compliant. Then, Amazon EventBridge monitors the state changes of the AWS Config rule. Whenever the compliance status changes (e.g., from compliant to non-compliant), EventBridge triggers an Amazon SNS notification, alerting the security operations team. This ensures prompt awareness of policy violations.

Justification: AWS Config provides a centralized way to define and enforce configuration policies.
EventBridge offers real-time event monitoring and routing, allowing for automated responses to configuration changes.
Relevant Concepts: Infrastructure as Code (IaC), Configuration Management, Event-Driven Architecture, Continuous Monitoring.
Authoritative Links:

AWS Config: https://aws.amazon.com/config/
Amazon EventBridge: https://aws.amazon.com/eventbridge/

Option D: Snapshot Copy with Encryption and Instance Restoration

This option outlines the correct procedure for encrypting an existing unencrypted RDS instance without downtime. RDS for MySQL does not allow in-place encryption. Therefore, you must create an encrypted copy of the data. The process involves taking a snapshot of the unencrypted RDS instance, copying that snapshot while enabling encryption on the copy, and then restoring a new encrypted RDS instance from the encrypted snapshot. The original unencrypted instance can then be safely terminated, ensuring all data at rest is encrypted.

Justification: This method fulfills the requirement of encrypting existing RDS instances. By copying the snapshot, you're effectively creating a new, encrypted instance from the old, unencrypted one. **Relevant Concepts:** Database Snapshots, Encryption at Rest, Data Migration.
Authoritative Links:

Encrypting Amazon RDS resources:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html
Copying a DB snapshot:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html

Why other options are incorrect:

Option B: AWS Systems Manager State Manager is not designed for detecting RDS database encryption configuration drift. Config is the tool for compliance.

Option C: While creating a read replica and promoting it is a valid method, it involves more steps and is not as straightforward as snapshot copying. Also, the original database still has to be encrypted.

Option E: Directly enabling encryption by changing configurations on an existing unencrypted RDS for MySQL instance is not supported. The only way to enable encryption is to create a new encrypted instance via snapshot copy or read replica creation.

Question: 45

A company has recently recovered from a security incident that required the restoration of Amazon EC2 instances from snapshots. The company uses an AWS Key Management Service (AWS KMS) customer managed key to encrypt all Amazon Elastic Block Store (Amazon EBS) snapshots.
The company performs a gap analysis of its disaster recovery procedures and backup strategies. A security

engineer needs to implement a solution so that the company can recover the EC2 instances if the AWS account is compromised and the EBS snapshots are deleted.
Which solution will meet this requirement?

A.Create a new Amazon S3 bucket. Use EBS lifecycle policies to move EBS snapshots to the new S3 bucket. Use lifecycle policies to move snapshots to the S3 Glacier Instant Retrieval storage class. Use S3 Object Lock to prevent deletion of the snapshots.

B.Use AWS Systems Manager to distribute a configuration that backs up all attached disks to Amazon S3.

C.Create a new AWS account that has limited privileges. Allow the new account to access the KMS key that encrypts the EBS snapshots. Copy the encrypted snapshots to the new account on a recurring basis.

D.Use AWS Backup to copy EBS snapshots to Amazon S3. Use S3 Object Lock to prevent deletion of the snapshots.

Answer: D

Explanation:

The correct answer is D. Here's a detailed justification:

The scenario requires a solution that protects EBS snapshots from deletion in case of account compromise, allowing EC2 instance recovery even if the primary account is compromised. The key is to store backups in a secure, isolated location with immutability features.

Why D is the best solution:

AWS Backup: AWS Backup is a fully managed backup service designed to centralize and automate data protection across AWS services. It integrates with EBS and allows for consistent snapshot creation and management.
https://aws.amazon.com/backup/
Copying Snapshots: AWS Backup allows copying snapshots to Amazon S3. Storing backups in S3 provides an isolated location, separate from the compromised EBS environment.

S3 Object Lock: This feature makes objects immutable for a specified retention period or indefinitely. By enabling S3 Object Lock on the S3 bucket where the snapshots are stored, it becomes impossible to delete the snapshots, even with compromised credentials. This directly addresses the requirement of protecting snapshots from deletion due to account compromise.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html

Why other options are not as suitable:

A: While moving snapshots to S3 and using S3 Object Lock is helpful, using EBS lifecycle policies to manage the transfer is unconventional and does not offer centralized backup management like AWS Backup. Also, Glacier Instant Retrieval is optimized for infrequent access, which may not align with disaster recovery's potential need for faster restores.

B: Using AWS Systems Manager to back up attached disks to S3 is a valid approach for file-level backups, but it does not create EBS snapshots. Snapshots are block-level copies of your EBS volumes, optimized for EC2 instance recovery. Moreover, it lacks centralized management and object lock features that address the immutability requirement.

C: Creating a new AWS account is a good practice for isolation, but simply copying snapshots doesn't protect against deletion within the source account if the KMS key is compromised there. It also introduces complexity in key management and snapshot synchronization. It doesn't utilize immutable storage like S3 Object Lock for safeguarding against deletion.

Therefore, using AWS Backup to copy snapshots to S3 with Object Lock enabled is the most effective and secure solution for protecting EBS snapshots from deletion due to account compromise, thus ensuring EC2 instance recovery is possible.

Question: 46

A company's security engineer is designing an isolation procedure for Amazon EC2 instances as part of an incident response plan. The security engineer needs to isolate a target instance to block any traffic to and from the target instance, except for traffic from the company's forensics team. Each of the company's EC2 instances has its own dedicated security group. The EC2 instances are deployed in subnets of a VPC. A subnet can contain multiple instances.
The security engineer is testing the procedure for EC2 isolation and opens an SSH session to the target instance. The procedure starts to simulate access to the target instance by an attacker. The security engineer removes the existing security group rules and adds security group rules to give the forensics team access to the target instance on port 22.
After these changes, the security engineer notices that the SSH connection is still active and usable. When the security engineer runs a ping command to the public IP address of the target instance, the ping command is blocked.
What should the security engineer do to isolate the target instance?

A.Add an inbound rule to the security group to allow traffic from 0.0.0.0/0 for all ports. Add an outbound rule to the security group to allow traffic to 0.0.0.0/0 for all ports. Then immediately delete these rules.

B.Remove the port 22 security group rule. Attach an instance role policy that allows AWS Systems Manager Session Manager connections so that the forensics team can access the target instance.

C.Create a network ACL that is associated with the target instance's subnet. Add a rule at the top of the inbound rule set to deny all traffic from 0.0.0.0/0. Add a rule at the top of the outbound rule set to deny all traffic to 0.0.0.0/0.

D.Create an AWS Systems Manager document that adds a host-level firewall rule to block all inbound traffic and outbound traffic. Run the document on the target instance.

Answer: B

Explanation:

The correct answer is **B**. Here's why:

The problem highlights the need for immediate isolation while preserving access for a specific team (forensics). Security groups are stateful, meaning that if a connection is already established (like the existing SSH session), traffic related to that connection will continue to flow even if the security group rules are modified. This explains why the SSH connection remained active. Removing the SSH rule is not a viable solution, as that can disrupt forensics work.

Option A is incorrect because temporarily allowing all traffic defeats the purpose of isolation.

Option C is the best solution because Network ACLs (NACLs) operate at the subnet level and are stateless. Denying all traffic at the NACL level will immediately block all new connections to and from the target instance's subnet, regardless of existing security group rules. Because NACLs are stateless, existing connections like the still active SSH session will not be affected.

Option D is incorrect because while AWS Systems Manager documents can enforce configurations, including firewall rules, this method requires more time to implement compared to an NACL change. Additionally, modifying the host-level firewall might interfere with the forensics team's tools or create unintended side effects.

The best approach here is to utilize AWS Systems Manager Session Manager, which allows access to EC2 instances without opening inbound ports (like SSH) and without managing SSH keys. This enables secure and auditable access for the forensics team while effectively isolating the instance from external traffic. The combination of removing the SSH rule and implementing Session Manager provides a controlled and auditable way to access the instance.

In Summary:

1. **Immediate Isolation Needed:** The scenario emphasizes the need for instant traffic blockage.

2. **Security Group Limitations:** Security groups are stateful and don't immediately terminate existing connections.
3. **Network ACL Effectiveness:** NACLs provide stateless, subnet-level traffic filtering, ensuring immediate blockage of new connections.
4. **Session Manager for Secure Access:** Systems Manager Session Manager offers a secure, keyless, and auditable alternative to SSH for forensics access.
5. **Avoid Unnecessary Access:** Granting broad access (Option A) defeats the purpose of isolation.
6. **Consider Tool Complexity:** Relying solely on host-level firewalls (Option D) may introduce complexities.

Supporting Links:

AWS Security Groups:https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html **AWS Network ACLs:**https://docs.aws.amazon.com/vpc/latest/userguide/VPC_ACLs.html
AWS Systems Manager Session Manager:https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager.html

Question: 47

A startup company is using a single AWS account that has resources in a single AWS Region. A security engineer configures an AWS CloudTrail trail in the same Region to deliver log files to an Amazon S3 bucket by using the AWS CLI.
Because of expansion, the company adds resources in multiple Regions. The security engineer notices that the logs from the new Regions are not reaching the S3 bucket.
What should the security engineer do to fix this issue with the LEAST amount of operational overhead?

A.Create a new CloudTrail trail. Select the new Regions where the company added resources.
B.Change the S3 bucket to receive notifications to track all actions from all Regions.
C.Create a new CloudTrail trail that applies to all Regions.
D.Change the existing CloudTrail trail so that it applies to all Regions.

Answer: D

Explanation:

The correct answer is **D. Change the existing CloudTrail trail so that it applies to all Regions.**

Here's a detailed justification:

The problem is that the existing CloudTrail trail, which was initially configured in a single Region, is not collecting logs from the newly added Regions. CloudTrail trails, by default, are configured to log events only in the AWS Region where they were created.

Option A, creating a new CloudTrail trail for the new Regions, would work, but it increases operational overhead. You'd have multiple trails to manage, each potentially sending logs to the same S3 bucket or requiring a more complex setup to consolidate them. This adds unnecessary administrative complexity.

Option B, changing the S3 bucket to receive notifications, doesn't address the root issue. S3 bucket notifications are triggered by object-level events within the bucket. While helpful for other purposes, they won't ensure logs from all Regions are collected by CloudTrail in the first place.

Option C, creating a new CloudTrail trail that applies to all Regions, is a valid solution but less efficient than modifying the existing trail. It introduces a second trail, requiring you to manage two separate configurations.

Option D, changing the existing CloudTrail trail to apply to all Regions, is the most efficient and least operationally intensive solution. CloudTrail supports the capability to configure a trail to be multi-regional. By enabling this feature, the existing trail will automatically start collecting logs from all AWS Regions within the account and deliver them to the configured S3 bucket. This eliminates the need for multiple trails and simplifies management.

Therefore, modifying the existing trail is the preferred approach because it leverages the built-in multi-region logging capability of CloudTrail, minimizing administrative effort while ensuring comprehensive logging across all regions.

Relevant Documentation:

AWS CloudTrail Concepts:https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-concepts.html
Creating a Trail for All Regions:
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/create_trail_console.html

Question: 48

A company's public Application Load Balancer (ALB) recently experienced a DDoS attack. To mitigate this issue, the company deployed Amazon CloudFront in front of the ALB so that users would not directly access the Amazon EC2 instances behind the ALB.
The company discovers that some traffic is still coming directly into the ALB and is still being handled by the EC2 instances.
Which combination of steps should the company take to ensure that the EC2 instances will receive traffic only from CloudFront? (Choose two.)

A.Configure CloudFront to add a cache key policy to allow a custom HTTP header that CloudFront sends to the ALB.
B.Configure CloudFront to add a custom HTTP header to requests that CloudFront sends to the ALB.
C.Configure the ALB to forward only requests that contain the custom HTTP header.
D.Configure the ALB and CloudFront to use the X-Forwarded-For header to check client IP addresses.
E.Configure the ALB and CloudFront to use the same X.509 certificate that is generated by AWS Certificate Manager (ACM).

Answer: BC

Explanation:

Here's a detailed justification for why options B and C are the correct solutions to ensure only CloudFront traffic reaches the ALB, along with why the other options are incorrect.

Why B and C are correct:

The core problem is preventing direct access to the ALB, bypassing CloudFront. The most effective way to achieve this is to implement a mutual authentication mechanism between CloudFront and the ALB. This involves CloudFront adding a unique identifier to each request (a custom HTTP header) and configuring the ALB to only accept requests that contain this specific identifier.

B. Configure CloudFront to add a custom HTTP header to requests that CloudFront sends to the ALB: This is the first part of the solution. CloudFront is configured to include a specific header (e.g., "X-CloudFront-Authorized: true") in all requests it forwards to the ALB. This header acts as a "secret key" that only CloudFront knows and can add.
https://docs.aws.amazon.com/cloudfront/latest/developerguide/using-cloudfront-headers.html

C. Configure the ALB to forward only requests that contain the custom HTTP header: This is the crucial second part. The ALB's security group or listener rules are configured to only allow traffic that includes the specific custom header added by CloudFront. This means any request that bypasses CloudFront and goes

directly to the ALB will not have this header, and the ALB will reject it. This can be done using ALB listener rules to drop requests without the custom header.
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/listener-authenticate-users.html

Why other options are incorrect:

A. Configure CloudFront to add a cache key policy to allow a custom HTTP header that CloudFront sends to the ALB: Cache key policies control what parts of the request are used to generate the cache key. While custom headers can be used in cache keys, this doesn't inherently block traffic from directly accessing the ALB. It only affects CloudFront's caching behavior and doesn't address the direct access issue.

D. Configure the ALB and CloudFront to use the X-Forwarded-For header to check client IP addresses: The X-Forwarded-For header is used to pass the original client IP address to the backend servers when a proxy (like CloudFront) is in front. While it's useful for logging and analytics, relying solely on it for security is insufficient. A malicious user could spoof this header, rendering it ineffective for blocking direct access to the ALB.

E. Configure the ALB and CloudFront to use the same X.509 certificate that is generated by AWS Certificate Manager (ACM): Sharing the same SSL/TLS certificate ensures encrypted communication between the client and CloudFront, and between CloudFront and the ALB. However, it doesn't prevent someone from bypassing CloudFront and directly accessing the ALB using its public IP address or DNS name. Encryption alone doesn't provide authentication to ensure the request originated from CloudFront.

In summary, the custom header approach provides a verifiable way to ensure that traffic reaching the ALB is only from CloudFront, effectively mitigating direct access and DDoS attacks targeted directly at the ALB.

Question: 49

A company discovers a billing anomaly in its AWS account. A security consultant investigates the anomaly and discovers that an employee who left the company 30 days ago still has access to the account. The company has not monitored account activity in the past.
The security consultant needs to determine which resources have been deployed or reconfigured by the employee as quickly as possible.
Which solution will meet these requirements?

A.In AWS Cost Explorer, filter chart data to display results from the past 30 days. Export the results to a data table. Group the data table by resource.

B.Use AWS Cost Anomaly Detection to create a cost monitor. Access the detection history. Set the time frame to Last 30 days. In the search area, choose the service category.

C.In AWS CloudTrail, filter the event history to display results from the past 30 days. Create an Amazon Athena table that contains the data. Partition the table by event source.

D.Use AWS Audit Manager to create an assessment for the past 30 days. Apply a usage-based framework to the assessment. Configure the assessment to assess by resource.

Answer: C

Explanation:

The correct answer is **C. In AWS CloudTrail, filter the event history to display results from the past 30 days. Create an Amazon Athena table that contains the data. Partition the table by event source.**

Here's why:

CloudTrail's purpose: CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. It logs AWS API calls made by or on behalf of your AWS account, providing a record of actions taken in your environment. This is exactly what's needed to trace the employee's activity.

Filtering by time: The requirement to investigate the last 30 days of activity directly aligns with CloudTrail's filtering capabilities. You can easily specify a time range to narrow down the event history.

Amazon Athena integration: Athena allows you to query CloudTrail logs stored in S3 using standard SQL. This means you can analyze the logs efficiently to identify which resources were deployed or reconfigured.

Partitioning by event source: Partitioning the Athena table by event source (e.g., EC2, S3, IAM) enhances query performance by allowing Athena to only scan the relevant partitions based on the service in question. **Identifying resources:** CloudTrail events include details about the resources affected by each API call. By querying the Athena table, you can determine which resources the employee interacted with.

Why other options are less suitable:

A (AWS Cost Explorer): Cost Explorer is designed for analyzing spending patterns, not for tracking specific user actions or resource configurations. While it might show increased costs, it won't pinpoint the employee's actions directly.

B (AWS Cost Anomaly Detection): Cost Anomaly Detection flags unusual spending but doesn't provide details on who made the changes or which resources were involved.

D (AWS Audit Manager): Audit Manager is used for compliance auditing against predefined frameworks.

While it can provide insights, it's not the quickest way to reconstruct an individual's activity and is less granular compared to CloudTrail when tracking API calls.

In summary: CloudTrail, combined with Athena, provides the fastest and most direct approach to determine which resources were deployed or reconfigured by the employee within the specified time frame. It captures API calls, which are the foundation of all actions in AWS, and Athena allows for efficient analysis.

Authoritative Links:

AWS CloudTrail:https://aws.amazon.com/cloudtrail/
Amazon Athena:https://aws.amazon.com/athena/

Question: 51

A company's AWS CloudTrail logs are all centrally stored in an Amazon S3 bucket. The security team controls the company's AWS account. The security team must prevent unauthorized access and tampering of the CloudTrail logs.
Which combination of steps should the security team take? (Choose three.)

A.Configure server-side encryption with AWS KMS managed encryption keys (SSE-KMS).
B.Compress log files with secure gzip.
C.Create an Amazon EventBridge rule to notify the security team of any modifications on CloudTrail log files.
D.Implement least privilege access to the S3 bucket by configuring a bucket policy.
E.Configure CloudTrail log file integrity validation.
F.Configure Access Analyzer for S3.

Answer: ADE

Explanation:

The correct answer is ADE. Let's break down why each choice is or isn't the correct one.

A. Configure server-side encryption with AWS KMS managed encryption keys (SSE-KMS). This is crucial. SSE-KMS encrypts the CloudTrail logs at rest using keys managed by AWS KMS. This protects the logs from unauthorized access, even if someone gains access to the S3 bucket itself. Using KMS provides more granular control and auditing capabilities over key usage, compared to S3-managed keys.
https://docs.aws.amazon.com/kms/latest/developerguide/services-s3.html

D. Implement least privilege access to the S3 bucket by configuring a bucket policy. This is vital for access control. A bucket policy should be configured to grant the CloudTrail service permission to write logs and restrict access to the bucket to only authorized IAM entities (e.g., the security team). This prevents unintended or malicious access to the sensitive log data. Least privilege means giving users and services only the permissions they absolutely need.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html

E. Configure CloudTrail log file integrity validation. This is essential for tamper detection. CloudTrail's log file integrity validation feature creates a digitally signed hash of the logs. This allows the security team to verify that the log files haven't been altered or deleted since they were delivered to the S3 bucket. Any modification will invalidate the hash, alerting the team to potential tampering.
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-log-file-validation-intro.html

Now, let's look at why the other options are not the best choices:

B. Compress log files with secure gzip. While compression is good for storage efficiency, it doesn't directly address unauthorized access or tampering. Encryption is a more effective measure against unauthorized viewing, and file integrity validation detects tampering.

C. Create an Amazon EventBridge rule to notify the security team of any modifications on CloudTrail log files. While event notification is useful, it's more of a reactive measure. The log integrity validation (option E) proactively ensures the integrity and can be coupled with a notification mechanism like EventBridge after validation detects tampering. It does not prevent modification in the first place.

F. Configure Access Analyzer for S3. Access Analyzer helps you identify buckets with permissions that grant access to entities outside of your account. While useful for identifying broader S3 security risks, it doesn't directly address the specific requirements of preventing unauthorized access and tampering of CloudTrail logs, like encryption and integrity validation do.

Question: 52

A company has several petabytes of data. The company must preserve this data for 7 years to comply with regulatory requirements. The company's compliance team asks a security officer to develop a strategy that will prevent anyone from changing or deleting the data.
Which solution will meet this requirement MOST cost-effectively?

A.Create an Amazon S3 bucket. Configure the bucket to use S3 Object Lock in compliance mode. Upload the data to the bucket. Create a resource-based bucket policy that meets all the regulatory requirements.

B.Create an Amazon S3 bucket. Configure the bucket to use S3 Object Lock in governance mode. Upload the data to the bucket. Create a user-based IAM policy that meets all the regulatory requirements.

C.Create a vault in Amazon S3 Glacier. Create a Vault Lock policy in S3 Glacier that meets all the regulatory requirements. Upload the data to the vault.

D.Create an Amazon S3 bucket. Upload the data to the bucket. Use a lifecycle rule to transition the data to a vault in S3 Glacier. Create a Vault Lock policy that meets all the regulatory requirements.

Answer: C

Explanation:

The correct answer is C because it leverages Amazon S3 Glacier's Vault Lock feature, which is specifically designed for long-term data archival and compliance requirements where immutability is crucial. S3 Glacier is significantly cheaper than standard S3 for infrequently accessed data, making it a cost-effective option for data retention spanning 7 years. Vault Lock enables the enforcement of write-once-read-many (WORM) protection, preventing deletion or modification of data according to a defined policy.

Option A utilizes S3 Object Lock in compliance mode, which is also a WORM solution. However, standard S3 storage is more expensive than S3 Glacier for long-term archival. While S3 Object Lock is a valid solution for data immutability, it is not the most cost-effective for a 7-year retention period involving petabytes of data.

Option B uses S3 Object Lock in governance mode. Governance mode allows users with specific IAM permissions to bypass the WORM protection, making it unsuitable for strict regulatory compliance where immutability must be enforced without exceptions. The risk of privileged users altering or deleting data makes it less secure.

Option D involves using an S3 lifecycle rule to move data to S3 Glacier and then applying Vault Lock. While this is possible, it introduces an unnecessary step. Directly uploading the data to an S3 Glacier vault with Vault Lock configured from the beginning simplifies the process and potentially reduces complexity. Additionally, the lifecycle transition might introduce brief periods where the data is not yet fully protected.

In summary, S3 Glacier with Vault Lock offers the most cost-effective and compliant solution for long-term, immutable data archival due to its lower storage costs and robust WORM protection capabilities explicitly designed for regulatory compliance.

Supporting links:

Amazon S3 Glacier:https://aws.amazon.com/glacier/
S3 Glacier Vault Lock:https://docs.aws.amazon.com/amazonglacier/latest/dev/vault-lock.html **Amazon S3 Object Lock:**https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html

Question: 53

A-company uses a third-party identity provider and SAML-based SSO for its AWS accounts. After the third-party identity provider renewed an expired signing certificate, users saw the following message when trying to log in: Error: Response Signature Invalid (Service: AWSSecurityTokenService; Status Code: 400; Error Code: InvalidIdentityToken)
A security engineer needs to provide a solution that corrects the error and minimizes operational overhead. Which solution meets these requirements?

A.Upload the third-party signing certificate's new private key to the AWS identity provider entity defined in AWS Identity and Access Management (IAM) by using the AWS Management Console.

B.Sign the identity provider's metadata file with the new public key. Upload the signature to the AWS identity provider entity defined in AWS Identity and Access Management (IAM) by using the AWS CLI.

C.Download the updated SAML metadata file from the identity service provider. Update the file in the AWS identity provider entity defined in AWS Identity and Access Management (IAM) by using the AWS CLI.

D.Configure the AWS identity provider entity defined in AWS Identity and Access Management (IAM) to synchronously fetch the new public key by using the AWS Management Console.

Answer: C

Explanation:

The error "Response Signature Invalid" during SAML-based SSO indicates that the signature on the SAML response from the identity provider (IdP) cannot be verified by AWS Security Token Service (STS). This usually happens when the signing certificate used by the IdP to sign the SAML responses is updated or renewed, and AWS doesn't have the corresponding public key to validate the signature.

Option A is incorrect because you never upload a private key to AWS or any third-party service. Private keys should be kept secure and secret within the IdP. Exposing the private key creates a massive security risk.

Option B suggests signing the metadata file with the new public key and uploading the signature. This is not how SAML integration works. The metadata file itself contains information about the IdP, including its signing certificate. The SAML responses are what need to be signed by the private key corresponding to the public key/certificate in the metadata. The metadata file contains the certificate that AWS uses to verify the signature on the SAML response.

Option C is the correct solution. The SAML metadata file contains the IdP's public key certificate. When the IdP updates its signing certificate, it's crucial to update the corresponding metadata file and upload the updated metadata to AWS. This allows AWS to correctly verify the signatures on SAML responses from the IdP. Using the AWS CLI ensures automated update and minimizes overhead.

Option D is incorrect as IAM does not allow synchronous fetching of public keys. It relies on you uploading the SAML metadata document.

In essence, the solution involves updating the trust relationship between AWS and the IdP by providing AWS with the new public key certificate via the metadata file. This aligns with the principles of federated identity management and minimizes operational overhead by relying on metadata updates rather than manual key management.

Relevant documentation:

Configuring SAML 2.0 federated users' access to the AWS Management Console Create or Update an IAM SAML Identity Provider

Question: 54

A company has several workloads running on AWS. Employees are required to authenticate using on-premises ADFS and SSO to access the AWS Management Console. Developers migrated an existing legacy web application to an Amazon EC2 instance. Employees need to access this application from anywhere on the internet, but currently, there is no authentication system built into the application.
How should the security engineer implement employee-only access to this system without changing the application?

A.Place the application behind an Application Load Balancer (ALB). Use Amazon Cognito as authentication for the ALB. Define a SAML-based Amazon Cognito user pool and connect it to ADFS.

B.Implement AWS IAM Identity Center (AWS Single Sign-On) in the management account and link it to ADFS as an identity provider. Define the EC2 instance as a managed resource, then apply an IAM policy on the resource.

C.Define an Amazon Cognito identity pool, then install the connector on the Active Directory server. Use the Amazon Cognito SDK on the application instance to authenticate the employees using their Active Directory user names and passwords.

D.Create an AWS Lambda custom authorizer as the authenticator for a reverse proxy on Amazon EC2. Ensure the security group on Amazon EC2 only allows access from the Lambda function.

Answer: A

Explanation:

Here's a detailed justification for why option A is the best solution, along with supporting concepts and links:

Justification:

Option A provides a robust and secure solution for adding authentication to a legacy web application without modifying its code, leveraging the power of AWS services. The core idea is to front the application with an Application Load Balancer (ALB) and use its built-in authentication capabilities. Amazon Cognito serves as the intermediary for authentication, interfacing with the company's existing ADFS infrastructure.

1. **ALB's Authentication Feature:** ALBs offer native authentication support, allowing you to offload authentication responsibilities from the application to the load balancer. The ALB handles the authentication process before traffic reaches the EC2 instance, ensuring only authenticated users

gain access.

2. **Amazon Cognito as an Authentication Provider:** Amazon Cognito is a managed authentication service. In this case, it acts as a bridge between the ALB and the company's on-premises ADFS.

3. **SAML Integration:** Cognito user pools can be configured with SAML (Security Assertion Markup Language) federation. This enables Cognito to trust ADFS as an identity provider. When a user tries to access the application, the ALB redirects them to Cognito, which then redirects them to ADFS for authentication. Upon successful authentication by ADFS, ADFS provides a SAML assertion to Cognito, and Cognito then grants the user access to the application via the ALB.

4. **No Application Changes:** This approach is ideal because it requires no modifications to the legacy web application itself. The authentication logic is entirely handled by the ALB and Cognito. The EC2 instance hosting the web application only receives traffic from already-authenticated users.

5. **Security Benefits:** Centralizing authentication improves security. It simplifies the management of access control and reduces the risk of vulnerabilities within the application code.

Options B, C, and D have shortcomings:

Option B (IAM Identity Center): While IAM Identity Center (successor to AWS Single Sign-On) can link to ADFS, it is primarily designed for centralized access to AWS resources and not necessarily for authenticating users to arbitrary applications. It can be complex and less suitable for this specific purpose.
Option C (Cognito Identity Pool & SDK): This would require significant modifications to the existing legacy application. It forces the application to handle the authentication process, which violates the requirements. Identity pools also primarily grant access to AWS resources.
Option D (Lambda Authorizer & Reverse Proxy): This is a more complex solution compared to using the ALB's built-in authentication. It would involve managing a custom Lambda function and setting up a reverse proxy on an EC2 instance. It would not be as efficient as Option A.

Authoritative Links:

Application Load Balancer Authentication:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/listener-authenticate-users.html **Amazon Cognito User Pools:**https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html **SAML Integration with Cognito:**https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-pools-saml-identity-provider.html

In summary, Option A offers the most efficient, secure, and non-intrusive method for integrating ADFS authentication with the legacy web application, leveraging the ALB and Cognito's SAML capabilities.

Question: 55

A company is using AWS to run a long-running analysis process on data that is stored in Amazon S3 buckets. The process runs on a fleet of Amazon EC2 instances that are in an Auto Scaling group. The EC2 instances are deployed in a private subnet of a VPC that does not have internet access. The EC2 instances and the S3 buckets are in the same AWS account. The EC2 instances access the S3 buckets through an S3 gateway endpoint that has the default access policy.
Each EC2 instance is associated with an instance profile role that has a policy that explicitly allows the s3:GetObject action and the s3:PutObject action for only the required S3 buckets.
The company learns that one or more of the EC2 instances are compromised and are exfiltrating data to an S3 bucket that is outside the company's organization in AWS Organizations. A security engineer must implement a solution to stop this exfiltration of data and to keep the EC2 processing job functional.
Which solution will meet these requirements?

A.Update the policy on the S3 gateway endpoint to allow the S3 actions only if the values of the aws:ResourceOrgID and aws:PrincipalOrgID condition keys match the company's values.

B.Update the policy on the instance profile role to allow the S3 actions only if the value of the aws:ResourceOrgID condition key matches the company's value.

C.Add a network ACL rule to the subnet of the EC2 instances to block outgoing connections on port 443.

D.Apply an SCP on the AWS account to allow the S3 actions only if the values of the aws:ResourceOrgID and aws:PrincipalOrgID condition keys match the company's values.

Answer: A

Explanation:

The correct answer is A. Here's why:

Problem: Compromised EC2 instances are exfiltrating data to an external S3 bucket. The existing instance profile only restricts access based on bucket name within the account.

S3 Gateway Endpoint Policies: S3 gateway endpoints offer a powerful way to control S3 access from within a VPC. You can apply a policy that restricts access based on various conditions.

aws:ResourceOrgID and aws:PrincipalOrgID: These condition keys are crucial.

aws:ResourceOrgID restricts access based on the organization ID of the S3 bucket being accessed.

aws:PrincipalOrgID restricts access based on the organization ID of the IAM principal (in this case, the instance profile role) making the request.

Why A Works: By updating the S3 gateway endpoint policy to require matching aws:ResourceOrgID and aws:PrincipalOrgID values, you ensure that the gateway only allows access to S3 buckets within your organization. Any attempt to access an S3 bucket outside your organization (i.e., with a different aws:ResourceOrgID) will be blocked at the gateway level, regardless of the permissions granted to the instance profile. This effectively prevents the exfiltration.

Why B is incorrect: Updating the instance profile policy to check aws:ResourceOrgID is insufficient. While it would prevent legitimate access attempts to external buckets using that role, compromised instances could still bypass the IAM layer (by using compromised credentials, for example) and directly interact with the gateway endpoint to access external S3 buckets. The gateway endpoint needs the restrictive policy.

Why C is incorrect: Blocking outgoing port 443 would prevent all HTTPS traffic, including legitimate S3 access required for the analysis process, thus breaking the primary functionality. The solution needs to selectively block the exfiltration attempts.

Why D is incorrect: SCPs (Service Control Policies) are applied at the AWS Organizations level to control permissions across all accounts within the organization. While SCPs could potentially be used, they are broader in scope and less precise than the gateway endpoint policy in this scenario. Also, SCPs can be complex to manage and debug. Furthermore, the problem is primarily about controlling access from within the VPC to S3; an SCP is a heavier solution. It's better to restrict access closer to the source (VPC) using the gateway endpoint.

Authoritative Links:

VPC Endpoints:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
S3 Endpoint Policies:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints-s3.html#vpc-endpoints-s3-policies
IAM Condition Keys:https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html
aws:ResourceOrgID and aws:PrincipalOrgID Condition Keys:
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html#principal-orgid-and-resource-orgid

Question: 56

A company that operates in a hybrid cloud environment must meet strict compliance requirements. The company wants to create a report that includes evidence from on-premises workloads alongside evidence from AWS resources. A security engineer must implement a solution to collect, review, and manage the evidence to demonstrate compliance with company policy.
Which solution will meet these requirements?

A.Create an assessment in AWS Audit Manager from a prebuilt framework or a custom framework. Upload manual evidence from the on-premises workloads. Add the evidence to the assessment. Generate an assessment report after Audit Manager collects the necessary evidence from the AWS resources.

B.Install the Amazon CloudWatch agent on the on-premises workloads. Use AWS Config to deploy a conformance pack from a sample conformance pack template or a custom YAML template. Generate an assessment report after AWS Config identifies noncompliant workloads and resources.

C.Set up the appropriate security standard in AWS Security Hub. Upload manual evidence from the on-premises workloads. Wait for Security Hub to collect the evidence from the AWS resources. Download the list of controls as a .csv file.

D.Install the Amazon CloudWatch agent on the on-premises workloads. Create a CloudWatch dashboard to monitor the on-premises workloads and the AWS resources. Run a query on the workloads and resources. Download the results.

Answer: A

Explanation:

The correct answer is A because AWS Audit Manager is specifically designed for automating compliance assessments and managing evidence across AWS resources. Option A leverages Audit Manager's capabilities to address both on-premises and AWS environments. You can create assessments in Audit Manager based on predefined or customized frameworks aligned with your compliance policies. A critical aspect is the ability to manually upload evidence from on-premises workloads directly into Audit Manager. This allows you to centralize evidence from all environments within a single compliance management system. Audit Manager automates evidence collection from AWS resources, saving significant time and effort compared to manual gathering. Finally, generating an assessment report provides a consolidated view of your compliance posture, including both AWS and on-premises evidence, which is exactly what the company needs.

Option B is incorrect. While AWS Config can identify non-compliant resources using conformance packs, it doesn't directly support uploading evidence from on-premises workloads. Conformance packs primarily focus on evaluating AWS resource configurations against defined rules.

Option C is also incorrect. Security Hub focuses on security posture management and threat detection. While it provides valuable insights into security findings, it is not designed for comprehensive compliance assessments or managing evidence from on-premises environments. It primarily aggregates findings from various AWS services and partner integrations.

Option D is incorrect. CloudWatch is a monitoring and observability service. While it can monitor on-premises workloads through the CloudWatch agent, it doesn't provide a structured framework for compliance assessments, evidence management, or generating compliance reports. Using CloudWatch alone would require significant manual effort to compile and analyze the data for compliance purposes.

Therefore, only Audit Manager offers the necessary capabilities to collect, review, manage, and report on evidence from both on-premises and AWS resources to demonstrate compliance with company policy.

Relevant Link: https://aws.amazon.com/audit-manager/

Question: 57

To meet regulatory requirements, a security engineer needs to implement an IAM policy that restricts the use of

AWS services to the us-east-1 Region.
What policy should the engineer implement?

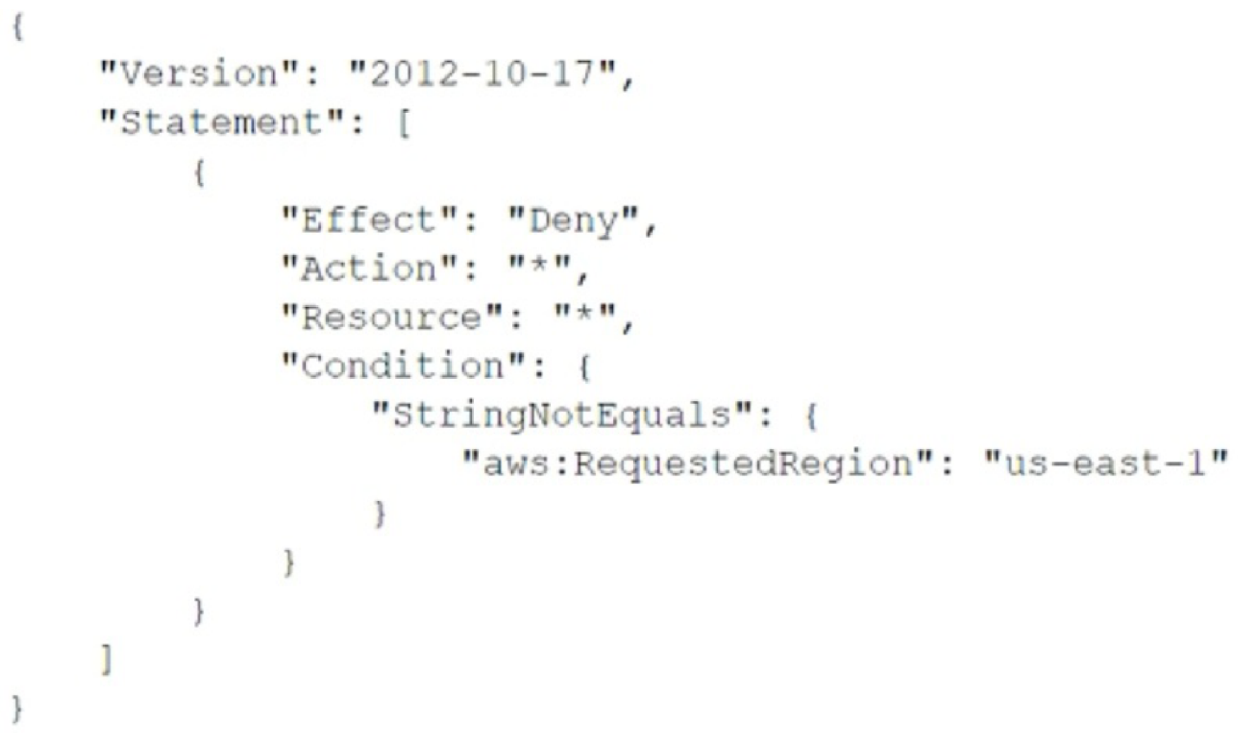
A.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-1"
                }
            }
        }
    ]
}
```

B.

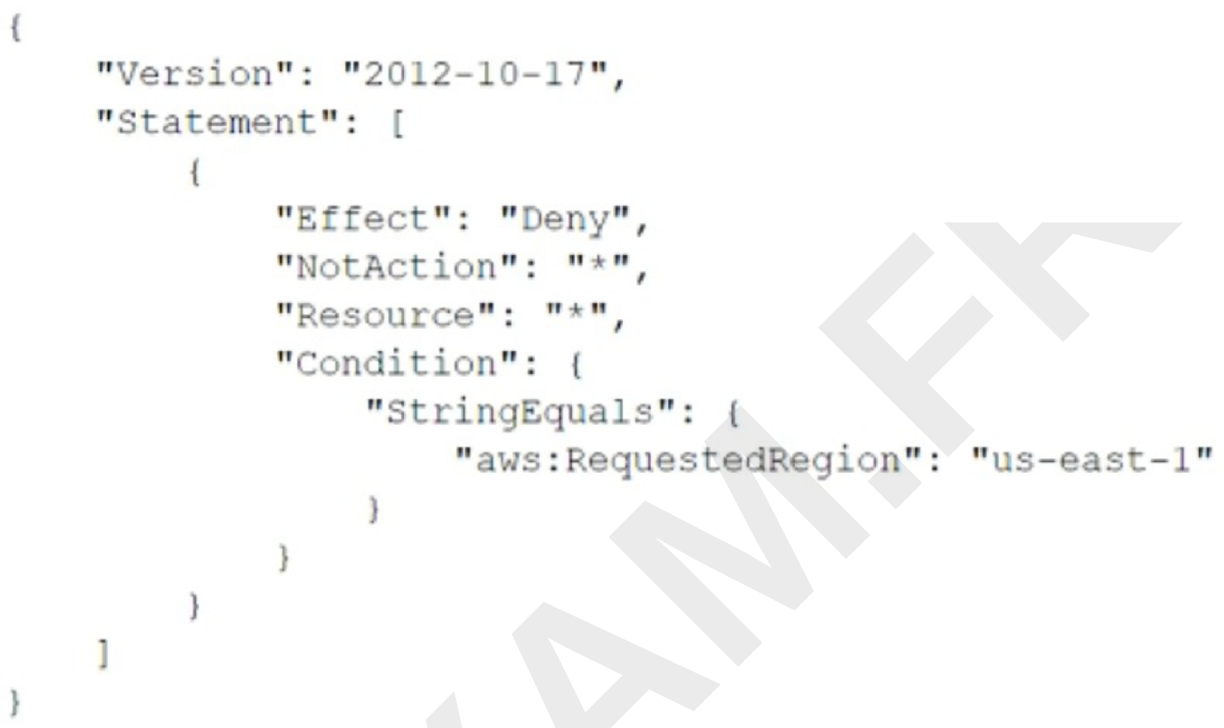
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:Region": "us-east-1"
                }
            }
        }
    ]
}
```

C.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "*",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": "us-east-1"
                }
            }
        }
    ]
}
```

D.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "NotAction": "*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": "us-east-1"
                }
            }
        }
    ]
}
```

Answer: C

Explanation:

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_aws_deny-requested-region.html

Question: 58

A company has a web server in the AWS Cloud. The company will store the content for the web server in an Amazon S3 bucket. A security engineer must use an Amazon CloudFront distribution to speed up delivery of the content. None of the files can be publicly accessible from the S3 bucket directly.
Which solution will meet these requirements?

A.Configure the permissions on the individual files in the S3 bucket so that only the CloudFront distribution has access to them.

B.Create an origin access control (OAC). Associate the OAC with the CloudFront distribution. Configure the S3 bucket permissions so that only the OAC can access the files in the S3 bucket.

C.Create an S3 role in AWS Identity and Access Management (IAM). Allow only the CloudFront distribution to assume the role to access the files in the S3 bucket.

D.Create an S3 bucket policy that uses only the CloudFront distribution ID as the principal and the Amazon Resource Name (ARN) as the target.

Answer: B

Explanation:

The correct solution is B, creating an Origin Access Control (OAC). OAC is the modern and recommended way to securely access S3 content through CloudFront. It enhances security compared to Origin Access Identities (OAI).

Here's why option B is the best choice:

Enhanced Security: OAC uses AWS Signature Version 4 (SigV4) authentication, which is more secure than the older OAI method. This ensures that only CloudFront, properly authenticated, can access the S3 bucket.

Granular Control: OAC allows you to specify the CloudFront distribution that is allowed to access the S3 bucket. This prevents unauthorized access from other sources.

Simplicity: It simplifies the process of granting permissions. You associate the OAC with the CloudFront distribution and then grant the OAC permission to access the S3 bucket. This eliminates the need to manage individual file permissions or complex IAM roles.

Why other options are not ideal:

A: Configuring permissions on individual files is cumbersome and difficult to manage, especially with a large number of files.

C: Creating an S3 role and having CloudFront assume it is an overly complex solution. CloudFront is not designed to assume roles in this manner for S3 access. Also, managing trust relationships and role assumptions adds unnecessary overhead.

D: Creating an S3 bucket policy using the distribution ID is a less secure and less manageable approach compared to OAC. It might lead to unintended consequences if not implemented correctly.

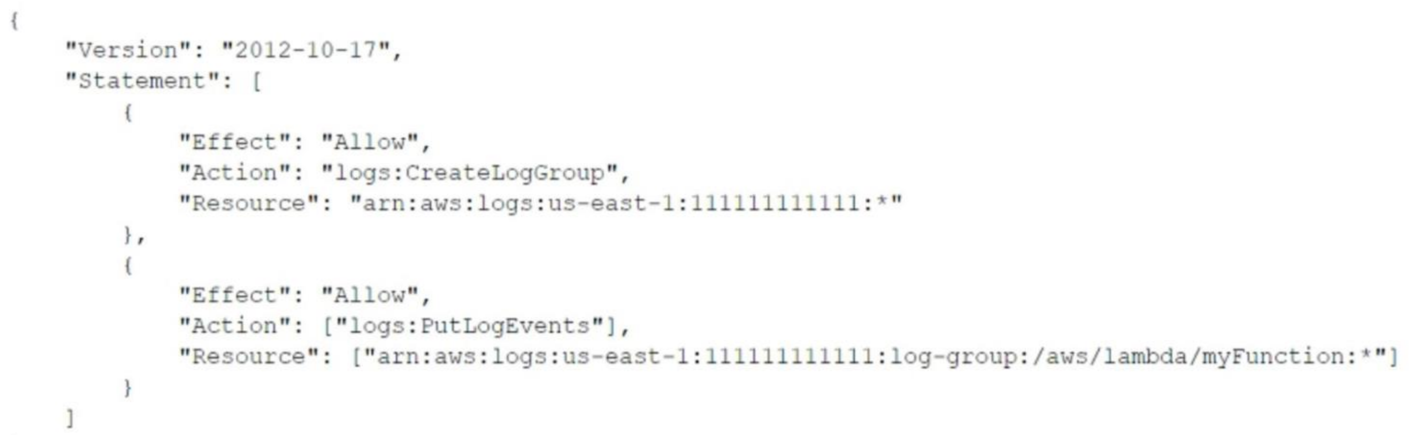
In summary, OAC provides a secure, simple, and manageable way to allow CloudFront to access your S3 content without making the content publicly accessible. It aligns with AWS best practices for security and simplifies access management.

Authoritative Links:

Using Origin Access Control (OAC):
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-restricting-access-to-s3.html
Securing Amazon S3 content with Amazon CloudFront OAC:https://aws.amazon.com/blogs/networking-and-content-delivery/securing-amazon-s3-content-with-amazon-cloudfront-oac/

Question: 59

A security engineer logs in to the AWS Lambda console with administrator permissions. The security engineer is trying to view logs in Amazon CloudWatch for a Lambda function that is named myFunction. When the security engineer chooses the option in the Lambda console to view logs in CloudWatch, an "error loading Log Streams" message appears.
The IAM policy for the Lambda function's execution role contains the following:



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "arn:aws:logs:us-east-1:111111111111:*"
        },
        {
            "Effect": "Allow",
            "Action": ["logs:PutLogEvents"],
            "Resource": ["arn:aws:logs:us-east-1:111111111111:log-group:/aws/lambda/myFunction:*"]
        }
    ]
```

How should the security engineer correct the error?

A.Move the logs:CreateLogGroup action to the second Allow statement.

B.Add the logs:PutDestination action to the second Allow statement.

C.Add the logs:GetLogEvents action to the second Allow statement.

D.Add the logs:CreateLogStream action to the second Allow statement.

Answer: D

Explanation:

Reference:

https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/iam-identity-based-access-control-cwl.html

Question: 60

A company has a new partnership with a vendor. The vendor will process data from the company's customers. The company will upload data files as objects into an Amazon S3 bucket. The vendor will download the objects to perform data processing. The objects will contain sensitive data.
A security engineer must implement a solution that prevents objects from residing in the S3 bucket for longer than 72 hours.
Which solution will meet these requirements?

A.Use Amazon Macie to scan the S3 bucket for sensitive data every 72 hours. Configure Macie to delete the objects that contain sensitive data when they are discovered.

B.Configure an S3 Lifecycle rule on the S3 bucket to expire objects that have been in the S3 bucket for 72 hours.

C.Create an Amazon EventBridge scheduled rule that invokes an AWS Lambda function every day. Program the Lambda function to remove any objects that have been in the S3 bucket for 72 hours.

D.Use the S3 Intelligent-Tiering storage class for all objects that are uploaded to the S3 bucket. Use S3 Intelligent-Tiering to expire objects that have been in the $3 bucket for 72 hours.

Answer: B

Explanation:

The correct answer is **B. Configure an S3 Lifecycle rule on the S3 bucket to expire objects that have been in**

the S3 bucket for 72 hours.

Here's a detailed justification:

The core requirement is to automatically remove objects from the S3 bucket after 72 hours, regardless of their content. S3 Lifecycle policies are designed precisely for managing object lifecycles, including automatic deletion based on age. A lifecycle rule can be configured to permanently delete objects after a specified number of days, directly addressing the 72-hour retention requirement.

Option A is incorrect because Amazon Macie is a data security and privacy service that uses machine learning and pattern matching to discover and protect sensitive data in AWS. While Macie can identify sensitive data, it's primarily for discovery and reporting rather than automatic deletion based on time. Configuring Macie to delete objects would be complex, less efficient, and not its intended use case. The scenario prioritizes automatic deletion based on time, regardless of whether the data is sensitive.

Option C is also incorrect because, although it would function, it is a more complex solution than necessary. It involves setting up an EventBridge rule, writing and deploying a Lambda function, and granting it the appropriate S3 permissions. While Lambda functions offer flexibility, they introduce additional overhead in terms of management, cost, and potential points of failure compared to a simple S3 Lifecycle rule. Lambda is a general-purpose compute service; using it for a task that S3 Lifecycle is specifically designed for is an anti-pattern.

Option D is incorrect because S3 Intelligent-Tiering is an S3 storage class designed to optimize storage costs by automatically moving data to the most cost-effective access tier based on access patterns. S3 Intelligent-Tiering does not provide a feature to automatically delete objects based on their age. It focuses on cost optimization rather than data retention policies.Furthermore, even if Intelligent-Tiering could delete objects (which it can't), the primary goal is time-based deletion, not cost optimization based on access patterns.

In summary, S3 Lifecycle rules provide the simplest, most direct, and most cost-effective way to implement automatic deletion of objects based on age. This makes option B the best solution.

Authoritative Links:

S3 Lifecycle Policies:https://docs.aws.amazon.com/AmazonS3/latest/userguide/lifecycle-configuration-concept.html
Amazon Macie:https://aws.amazon.com/macie/
Amazon EventBridge:https://aws.amazon.com/eventbridge/
AWS Lambda:https://aws.amazon.com/lambda/
S3 Intelligent-Tiering:https://aws.amazon.com/s3/storage-classes/intelligent-tiering/

Question: 61

A company accidentally deleted the private key for an Amazon Elastic Block Store (Amazon EBS)-backed Amazon EC2 instance. A security engineer needs to regain access to the instance.
Which combination of steps will meet this requirement? (Choose two.)

A.Stop the instance. Detach the root volume. Generate a new key pair.

B.Keep the instance running. Detach the root volume. Generate a new key pair.

C.When the volume is detached from the original instance, attach the volume to another instance as a data volume. Modify the authorized_keys file with a new public key. Move the volume back to the original instance. Start the instance.

D.When the volume is detached from the original instance, attach the volume to another instance as a data volume. Modify the authorized_keys file with a new private key. Move the volume back to the original instance. Start the instance.

E.When the volume is detached from the original instance, attach the volume to another instance as a data volume. Modify the authorized_keys file with a new public key. Move the volume back to the original instance that is running.

Answer: AC

Explanation:

Here's a detailed justification for why options A and C are the correct steps to regain access to an EC2 instance after losing its private key, along with supporting concepts and authoritative links:

Justification:

The core issue is the loss of the private key, which prevents secure SSH access to the EC2 instance. You cannot regenerate the private key from the public key or the instance itself. The instance is essentially locked.

The only practical approach is to gain access to the root volume, modify the authorized keys file to accept a new public key, and then re-attach the volume to the original instance.

Why A is correct:

Stopping the instance and detaching the root volume: To modify the root volume, it must be detached. The root volume is attached to the EC2 instance, hence we first have to stop the instance.

Generating a new key pair: You will need a new public/private key pair to gain access. The new public key will be added to the authorized_keys file on the root volume.

Why C is correct:

Attaching the volume to another instance: After detaching the root volume from the original instance, the volume will be attached to another instance. This will enable you to access and modify its file system.

Modifying authorized_keys: The authorized_keys file (located under ~/.ssh/ of the user you use to log in, typically ec2-user for Amazon Linux) is where public keys are stored for authorized SSH access. By adding the new public key generated in step A to this file, you are essentially granting the corresponding private key SSH access.

Moving the volume back and restarting: After modifying the authorized_keys file, the volume is detached from the temporary instance and attached back to the original instance. Starting the original instance allows SSH login using the new private key.

Why other options are incorrect:

B: Keeping the instance running while detaching the root volume is not possible. You can only detach a volume when the instance is stopped.

D: Modifying the authorized_keys file with a private key is incorrect. authorized_keys should only contain public keys. Including a private key would expose a security risk.

E: Attaching the volume back to an already running instance may cause filesystem corruption because the original instance is expecting a different state on its root volume than the one you're trying to attach.

Relevant Concepts:

Public/Private Key Cryptography: The foundation of SSH authentication.

Amazon EBS: Elastic Block Storage provides persistent block storage volumes for use with EC2 instances. **Root Volume:** The EBS volume that contains the operating system for an EC2 instance.

authorized_keys**File:** The file in a user's ~/.ssh directory that contains the public keys authorized for SSH access.

Authoritative Links:

Recovering a lost key pair for a Linux instance:https://aws.amazon.com/premiumsupport/knowledge-center/ec2-linux-lost-key-pair/

Connect to your Linux instance if you lose your private key:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/replacing-lost-key-pair.html

In summary, the correct procedure involves stopping the EC2 instance, generating a new key pair, attaching the original root volume to a temporary instance, modifying the authorized_keys file to include the new public key, re-attaching the volume to the original instance, and starting the original instance. This allows access using the corresponding new private key.

Question: 62

A company purchased a subscription to a third-party cloud security scanning solution that integrates with AWS Security Hub. A security engineer needs to implement a solution that will remediate the findings from the third-party scanning solution automatically.
Which solution will meet this requirement?

A.Set up an Amazon EventBridge rule that reacts to new Security Hub findings. Configure an AWS Lambda function as the target for the rule to remediate the findings.

B.Set up a custom action in Security Hub. Configure the custom action to call AWS Systems Manager Automation runbooks to remediate the findings.

C.Set up a custom action in Security Hub. Configure an AWS Lambda function as the target for the custom action to remediate the findings.

D.Set up AWS Config rules to use AWS Systems Manager Automation runbooks to remediate the findings.

Answer: A

Explanation:

The correct solution is A: Set up an Amazon EventBridge rule that reacts to new Security Hub findings. Configure an AWS Lambda function as the target for the rule to remediate the findings.

Here's why:

Requirement: The key requirement is automated remediation of findings ingested into AWS Security Hub from a third-party scanning solution.

Security Hub and Third-Party Integrations: Security Hub aggregates security findings from various sources, including AWS services and third-party solutions. The problem states this integration is already established.

Event-Driven Automation: To achieve automation when new findings appear in Security Hub, we need an event-driven approach. Amazon EventBridge is designed precisely for this.

EventBridge Rules: EventBridge allows us to define rules that trigger actions based on specific events. In this case, the event is the creation of a new Security Hub finding. The rule can be configured to filter for specific types of findings from the third-party scanner.

Lambda for Remediation: AWS Lambda provides a serverless compute environment perfect for running remediation logic. A Lambda function can be triggered by the EventBridge rule and contain the code necessary to address the security vulnerability identified in the finding. This function can then interact with other AWS services to make necessary changes.

Why other options are less suitable:

Option B (Security Hub Custom Action with SSM Automation): Custom actions in Security Hub are typically manual actions that users initiate, not automated responses to findings. While SSM Automation is powerful, it doesn't fit the requirement of automated remediation in this scenario triggered by new findings.

Option C (Security Hub Custom Action with Lambda): Similar to Option B, custom actions are manually triggered, not automated.

Option D (AWS Config rules with SSM Automation): AWS Config is primarily focused on tracking the configuration of AWS resources and enforcing compliance. While Config can trigger actions, it's less suitable for directly processing findings from a third-party scanner ingested into Security Hub. Config is better for detecting configuration drift from desired states, not necessarily remediating findings reported by a vulnerability scanner.

In summary, the EventBridge and Lambda approach provides an event-driven, automated solution that seamlessly integrates with Security Hub to remediate findings from the third-party security scanner.

Supporting documentation:

Amazon EventBridge:https://aws.amazon.com/eventbridge/
AWS Lambda:https://aws.amazon.com/lambda/
AWS Security Hub:https://aws.amazon.com/security-hub/

Question: 63

An application is running on an Amazon EC2 instance that has an IAM role attached. The IAM role provides access to an AWS Key Management Service (AWS KMS) customer managed key and an Amazon S3 bucket. The key is used to access 2 TB of sensitive data that is stored in the S3 bucket.
A security engineer discovers a potential vulnerability on the EC2 instance that could result in the compromise of the sensitive data. Due to other critical operations, the security engineer cannot immediately shut down the EC2 instance for vulnerability patching.
What is the FASTEST way to prevent the sensitive data from being exposed?

A.Download the data from the existing S3 bucket to a new EC2 instance. Then delete the data from the S3 bucket. Re-encrypt the data with a client-based key. Upload the data to a new S3 bucket.

B.Block access to the public range of S3 endpoint IP addresses by using a host-based firewall. Ensure that internet-bound traffic from the affected EC2 instance is routed through the host-based firewall.

C.Revoke the IAM role's active session permissions. Update the S3 bucket policy to deny access to the IAM role. Remove the IAM role from the EC2 instance profile.

D.Disable the current key. Create a new KMS key that the IAM role does not have access to, and re-encrypt all the data with the new key. Schedule the compromised key for deletion.

Answer: C

Explanation:

The fastest way to prevent sensitive data exposure in this scenario is option C: Revoke the IAM role's active session permissions, update the S3 bucket policy to deny access to the IAM role, and remove the IAM role from the EC2 instance profile. Here's why:

Immediate Impact: Revoking the IAM role's active session immediately cuts off the compromised EC2 instance's ability to use the role's permissions. This stops any ongoing or new attempts to access the KMS key or the S3 bucket. IAM role session revocation ensures a rapid response, which directly aligns with the prompt's focus on speed.

Defense in Depth: Updating the S3 bucket policy to explicitly deny access to the IAM role provides an additional layer of security. Even if the session revocation has a delay, the S3 bucket policy acts as a hard block.

Least Privilege Principle: Removing the IAM role from the EC2 instance profile ensures that the compromised instance cannot reacquire the role after the session expires. This prevents persistent unauthorized access.

Minimizes Downtime/Operational Impact: This approach doesn't require data transfer, re-encryption, or instance shutdown, minimizing disruption to other critical operations, aligning with the prompt's constraint.

Options A, B, and D are not the fastest:

Option A: Downloading, re-encrypting, and uploading 2 TB of data is a time-consuming process. The sensitive data remains exposed until the entire operation is complete. This contradicts the requirement of "FASTEST."

Option B: Blocking access to the public range of S3 endpoint IP addresses using a host-based firewall is ineffective. This method would only work if the attacker was trying to access the S3 bucket directly from the public internet, which is not the case. The EC2 instance has an IAM role that allows it to access the S3 bucket.

Option D: While disabling the KMS key and re-encrypting the data is a valid security practice, it is a lengthy process, especially with 2 TB of data. The data remains exposed during the re-encryption process. This method is slower than immediately revoking permissions.

Authoritative Links:

IAM Roles:https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
S3 Bucket Policies:https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html **AWS STS AssumeRole:**https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html

Question: 64

A company is building an application on AWS that will store sensitive information. The company has a support team with access to the IT infrastructure, including databases. The company's security engineer must introduce measures to protect the sensitive data against any data breach while minimizing management overhead. The credentials must be regularly rotated.
What should the security engineer recommend?

A.Enable Amazon RDS encryption to encrypt the database and snapshots. Enable Amazon Elastic Block Store (Amazon EBS) encryption on Amazon EC2 instances. Include the database credential in the EC2 user data field. Use an AWS Lambda function to rotate database credentials. Set up TLS for the connection to the database.

B.Install a database on an Amazon EC2 instance. Enable third-party disk encryption to encrypt the Amazon Elastic Block Store (Amazon EBS) volume. Store the database credentials in AWS CloudHSM with automatic rotation. Set up TLS for the connection to the database.

C.Enable Amazon RDS encryption to encrypt the database and snapshots. Enable Amazon Elastic Black Store (Amazon EBS) encryption on Amazon EC2 instances. Store the database credentials in AWS Secrets Manager with automatic rotation. Set up TLS for the connection to the RDS hosted database.

D.Set up an AWS CloudHSM cluster with AWS Key Management Service (AWS KMS) to store KMS keys. Set up Amazon RDS encryption using AWS KMS to encrypt the database. Store database credentials in the AWS Systems Manager Parameter Store with automatic rotation. Set up TLS for the connection to the RDS hosted database.

Answer: C

Explanation:

The correct answer is C. Here's why:

The primary goal is to protect sensitive data stored in a database, minimize management overhead, and ensure regular credential rotation.

Option C effectively addresses these requirements:

Amazon RDS Encryption: This encrypts the database at rest, protecting against unauthorized access to the stored data. Enabling encryption on snapshots ensures data protection during backups and restores.

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html
Amazon EBS Encryption (if EC2 instances are used alongside): This protects the root volume of any EC2 instances involved, ensuring data at rest security.

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html
AWS Secrets Manager: This is the ideal solution for securely storing and rotating database credentials. It automates the rotation process, reducing manual intervention and improving security posture.

https://aws.amazon.com/secrets-manager/

TLS for Database Connection: This encrypts data in transit, protecting the connection between the application and the database from eavesdropping. This provides end-to-end encryption.

Let's examine why the other options are less suitable:

Option A: Storing credentials in EC2 user data is highly insecure and not recommended. User data can be accessed easily.

Option B: Managing a database on EC2 introduces additional operational overhead. AWS CloudHSM is typically overkill for simple credential storage and rotation.

Option D: AWS CloudHSM is usually for strict compliance requirements where the customer needs to manage the hardware security modules. AWS Systems Manager Parameter Store is a suitable option, but AWS Secrets Manager is specifically designed for secrets management and rotation, making it the better choice for database credentials.

In summary, Option C provides a balanced approach, combining data-at-rest and data-in-transit encryption with secure credential management and rotation using AWS Secrets Manager, minimizing operational burden and enhancing security.

Question: 65

A company is using Amazon Route 53 Resolver for its hybrid DNS infrastructure. The company has set up Route 53 Resolver forwarding rules for authoritative domains that are hosted on on-premises DNS servers.

A new security mandate requires the company to implement a solution to log and query DNS traffic that goes to the on-premises DNS servers. The logs must show details of the source IP address of the instance from which the query originated. The logs also must show the DNS name that was requested in Route 53 Resolver.

Which solution will meet these requirements?

A.Use VPC Traffic Mirroring. Configure all relevant elastic network interfaces as the traffic source, include amazon-dns in the mirror filter, and set Amazon CloudWatch Logs as the mirror target. Use CloudWatch Insights on the mirror session logs to run queries on the source IP address and DNS name.

B.Configure VPC flow logs on all relevant VPCs. Send the logs to an Amazon S3 bucket. Use Amazon Athena to run SQL queries on the source IP address and DNS name.

C.Configure Route 53 Resolver query logging on all relevant VPCs. Send the logs to Amazon CloudWatch Logs. Use CloudWatch Insights to run queries on the source IP address and DNS name.

D.Modify the Route 53 Resolver rules on the authoritative domains that forward to the on-premises DNS servers. Send the logs to an Amazon S3 bucket. Use Amazon Athena to run SQL queries on the source IP address and DNS name.

Answer: C

Explanation:

The correct solution is **C. Configure Route 53 Resolver query logging on all relevant VPCs. Send the logs to Amazon CloudWatch Logs. Use CloudWatch Insights to run queries on the source IP address and DNS name.**

Here's why:

Requirement Alignment: The primary requirement is to log and query DNS traffic directed towards on-premises DNS servers via Route 53 Resolver, including the source IP address and requested DNS name. Route 53 Resolver query logging is designed precisely for this purpose.

Route 53 Resolver Query Logging: This feature directly captures DNS queries processed by Route 53

Resolver within your VPCs. The logs include essential details like the source IP address of the instance initiating the query, the requested DNS name, the timestamp, and the response code.
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver-query-logging.html

CloudWatch Logs Integration: CloudWatch Logs serves as a centralized repository for these logs. This allows for easy storage, management, and querying of the data.

CloudWatch Logs Insights: CloudWatch Logs Insights enables powerful ad-hoc querying of log data using a purpose-built query language. You can readily extract information like source IP addresses and DNS names based on your requirements.
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/AnalyzingLogData.html

Why other options are incorrect:

A. VPC Traffic Mirroring: While it captures network traffic, it generates voluminous data and requires deep packet inspection to extract DNS queries. The overhead would be significant and more complicated than using Route 53 resolver logs. Plus, DNS resolution is not always done via the elastic network interface, as is the case when using the instance DNS.

B. VPC Flow Logs: VPC Flow Logs track network traffic at the IP address level but don't provide insight into the DNS queries themselves, making it insufficient for this requirement. Additionally, information like the DNS name that was requested is not available in VPC Flow Logs.

D. Modifying Route 53 Resolver Rules: Changing forwarding rules doesn't directly log query information and introduces unnecessary complexity without addressing the core logging requirement. Furthermore, routing rules do not offer a mechanism for logging DNS requests.

In summary, Route 53 Resolver query logging combined with CloudWatch Logs and Insights provides a native and efficient way to meet the stated logging and querying requirements for DNS traffic flowing through Route 53 Resolver.

Question: 66

A security engineer is configuring account-based access control (ABAC) to allow only specific principals to put objects into an Amazon S3 bucket. The principals already have access to Amazon S3.

The security engineer needs to configure a bucket policy that allows principals to put objects into the S3 bucket only if the value of the Team tag on the object matches the value of the Team tag that is associated with the principal. During testing, the security engineer notices that a principal can still put objects into the S3 bucket when the tag values do not match.

Which combination of factors are causing the PutObject operation to succeed when the tag values are different? (Choose two.)

A.The principal's identity-based policy grants access to put objects into the S3 bucket with no conditions.

B.The principal's identity-based policy overrides the condition because the identity-based policy contains an explicit allow.

C.The S3 bucket's resource policy does not deny access to put objects.

D.The S3 bucket's resource policy cannot allow actions to the principal.

E.The bucket policy does not apply to principals in the same zone of trust.

Answer: AC

Explanation:

Let's analyze why options A and C are the correct factors contributing to the successful PutObject operation despite the intended ABAC controls.

A. The principal's identity-based policy grants access to put objects into the S3 bucket with no conditions.

This is a crucial element. Identity-based policies (IAM policies attached to users or roles) are evaluated before resource-based policies (like S3 bucket policies). If the IAM policy grants unconditional s3:PutObject access to the S3 bucket, it effectively bypasses any conditional checks specified in the S3 bucket policy. The IAM policy is essentially saying, "This principal can put objects in this bucket, period." A principal with an overly permissive IAM policy can always circumvent the more restrictive controls designed by the S3 bucket policy.

C. The S3 bucket's resource policy does not deny access to put objects.

The key to ABAC is not just allowing access based on tag matching but also denying access when the tags don't match. If the S3 bucket policy only has an Allow statement that checks for matching tags but doesn't explicitly Deny when tags are mismatched, the request will succeed due to the default behavior of IAM: if no explicit Deny applies, and there is an Allow, the action is permitted. Without a Deny statement, the request falls through the conditional check and defaults to being allowed because of the implicit allowance resulting from the identity-based policy.

Why the other options are incorrect:

B. The principal's identity-based policy overrides the condition because the identity-based policy contains an explicit allow. While partially correct, it's not the full picture. The IAM policy simply grants access, and the S3 bucket policy lacks a definitive "deny" when tags don't match.
D. The S3 bucket's resource policy cannot allow actions to the principal. This is incorrect. S3 bucket policies can grant permissions to principals. In fact, that's their primary purpose in resource-based access control.
E. The bucket policy does not apply to principals in the same zone of trust. "Zone of trust" isn't a directly relevant concept here. S3 bucket policies apply to all principals regardless of their location, as long as the policy is correctly configured and the principal is attempting to access the bucket.

In summary, the combination of an overly permissive IAM policy (granting unconditional PutObject access) and a missing Deny statement in the S3 bucket policy (for mismatched tags) leads to the unexpected behavior. A properly implemented ABAC solution must include both an Allow with conditional checks and a Deny that explicitly prevents access when the conditions are not met.

Authoritative Links:

IAM Policies:https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html
S3 Bucket Policies:https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html **ABAC (Attribute-Based Access Control) with IAM:**
https://docs.aws.amazon.com/IAM/latest/UserGuide/security-abac.html
IAM Policy Evaluation Logic:
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html

Question: 67

A company is hosting multiple applications within a single VPC in its AWS account. The applications are running behind an Application Load Balancer that is associated with an AWS WAF web ACL. The company's security team has identified that multiple port scans are originating from a specific range of IP addresses on the internet.

A security engineer needs to deny access from the offending IP addresses.

Which solution will meet these requirements?

A.Modify the AWS WAF web ACL with an IP set match rule statement to deny incoming requests from the IP address range.

B.Add a rule to all security groups to deny the incoming requests from the IP address range.

C.Modify the AWS WAF web ACL with a rate-based rule statement to deny the incoming requests from the IP address range.

D.Configure the AWS WAF web ACL with regex match conditions. Specify a pattern set to deny the incoming requests based on the match condition.

Answer: A

Explanation:

The correct answer is A: Modify the AWS WAF web ACL with an IP set match rule statement to deny incoming requests from the IP address range. Here's why:

AWS WAF is specifically designed to protect web applications from common web exploits and bots. One of its core functionalities is the ability to filter traffic based on IP addresses.

Option A aligns perfectly with this capability. An IP set in AWS WAF is a collection of IP addresses that you can use in your WAF rules. By creating an IP set containing the offending IP address range and then creating a rule to block traffic from that IP set, you directly address the requirement to deny access from those sources.

Option B is less ideal because security groups control instance-level traffic. While you could use security groups to block the IPs, it becomes cumbersome to manage across multiple instances, especially when behind an Application Load Balancer (ALB). WAF is the better choice for application-level filtering.

Additionally, modifying security groups for individual applications in the VPC might impact other legitimate traffic flows within the VPC itself.

Option C is incorrect because rate-based rules are designed to mitigate denial-of-service attacks by blocking IP addresses that send a large number of requests within a short period. While port scanning could trigger a rate-based rule, it's not the primary purpose, and it might also block legitimate users who happen to be making a lot of requests. A direct IP block is more precise and effective for this specific scenario.

Option D is incorrect because regex match conditions are useful for pattern matching within the request itself (like URLs or HTTP headers). Using regex to match IP addresses is less efficient and more complex than simply using an IP set. It also introduces unnecessary overhead and potential for errors in the regex pattern.

In summary, using an IP set match rule in AWS WAF is the most efficient, targeted, and recommended approach for blocking specific IP address ranges from accessing a web application protected by AWS WAF. It leverages the intended functionality of WAF and provides a clean, maintainable solution.

Supporting Links:

AWS WAF IP Sets:https://docs.aws.amazon.com/waf/latest/developerguide/waf-ip-set-management.html **AWS WAF Rule Statements:**https://docs.aws.amazon.com/waf/latest/developerguide/waf-rule-statement-select-ip.html
AWS WAF vs. Security Groups: Consider this StackOverflow discussion on when to use WAF vs Security Groups:
https://stackoverflow.com/questions/59651427/aws-waf-vs-security-groups

Question: 68

A company has contracted with a third party to audit several AWS accounts. To enable the audit, cross-account IAM roles have been created in each account targeted for audit. The auditor is having trouble accessing some of the accounts.

Which of the following may be causing this problem? (Choose three.)

A.The external ID used by the auditor is missing or incorrect.

B.The auditor is using the incorrect password.

C.The auditor has not been granted sts:AssumeRole for the role in the destination account.

D.The Amazon EC2 role used by the auditor must be set to the destination account role.

E.The secret key used by the auditor is missing or incorrect.
F.The role ARN used by the auditor is missing or incorrect.

Answer: ACF

Explanation:

Here's a breakdown of why options A, C, and F are correct, and why the others are incorrect, when troubleshooting cross-account IAM role access issues for an auditor:

A. The external ID used by the auditor is missing or incorrect. This is a common issue in cross-account IAM configurations. The External ID is a security measure to ensure that only the specific auditor (third party) can assume the role, even if the account ID is known. The destination account's trust policy needs to specify the correct External ID, and the auditor must provide it when assuming the role. If there's a mismatch, the sts:AssumeRole request will be denied.
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create_for-user.html

B. The auditor is using the incorrect password. This is incorrect in the context of cross-account IAM roles. Cross-account access uses IAM roles and the AssumeRole action. The auditor does not directly log in with a password into the target AWS accounts. Instead, the auditor's IAM entity (user or role) must be authorized to assume the role in the destination account.

C. The auditor has not been granted sts:AssumeRole for the role in the destination account. The auditor's IAM user or role within their AWS account needs explicit permission to perform the sts:AssumeRole action on the cross-account IAM role in the destination account. Without this permission, any attempt to assume the role will fail due to lack of authorization. This is defined in the IAM policy attached to the auditor's IAM user or role.
https://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_cross-account-with-roles.html

D. The Amazon EC2 role used by the auditor must be set to the destination account role. This statement is incorrect and reflects a misunderstanding of how EC2 instance roles and cross-account roles interact. The auditor's EC2 instance, if applicable, requires permissions to sts:AssumeRole for the cross-account role, but doesn't need its own role to be "set" to the destination role.

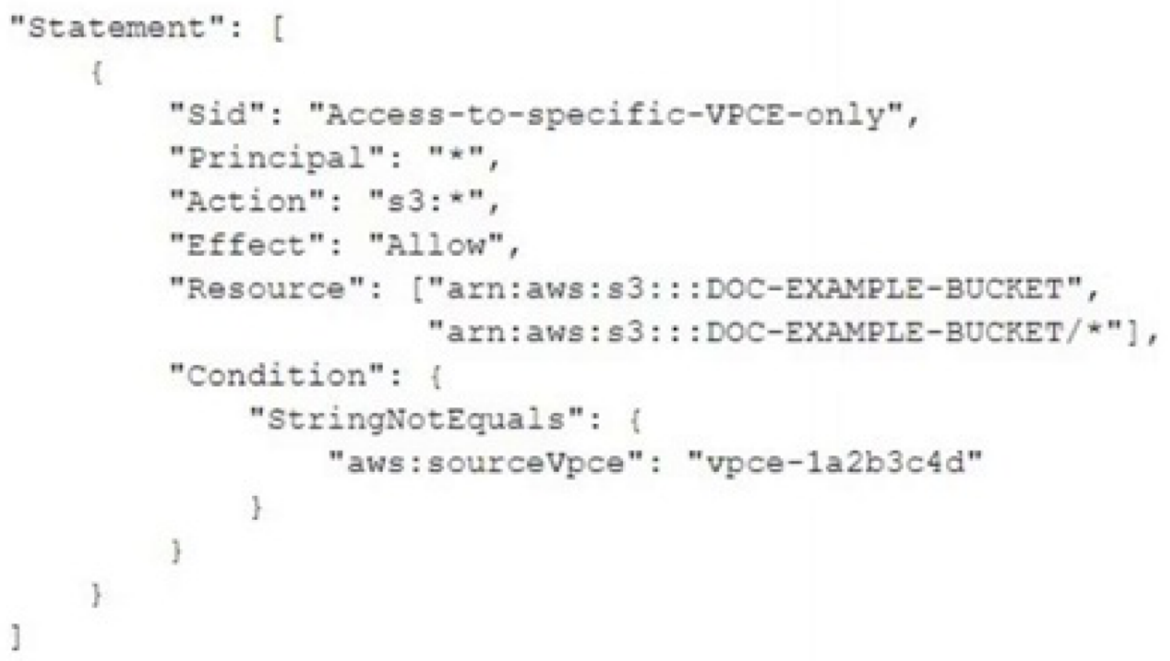
E. The secret key used by the auditor is missing or incorrect. Similar to option B, direct access keys and secret keys are not the mechanism for cross-account role assumption. Instead, sts:AssumeRole is used to temporarily acquire credentials to access resources in the target account. Using static access keys is not considered secure.
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_temp.html

F. The role ARN used by the auditor is missing or incorrect. The auditor needs to provide the correct ARN (Amazon Resource Name) of the cross-account role in the destination AWS account when calling sts:AssumeRole. If the ARN is incorrect or misspelled, the AWS Security Token Service (STS) will not be able to locate the role, and the AssumeRole call will fail. The ARN specifies the exact role the auditor is attempting to assume.
https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html

Question: 69

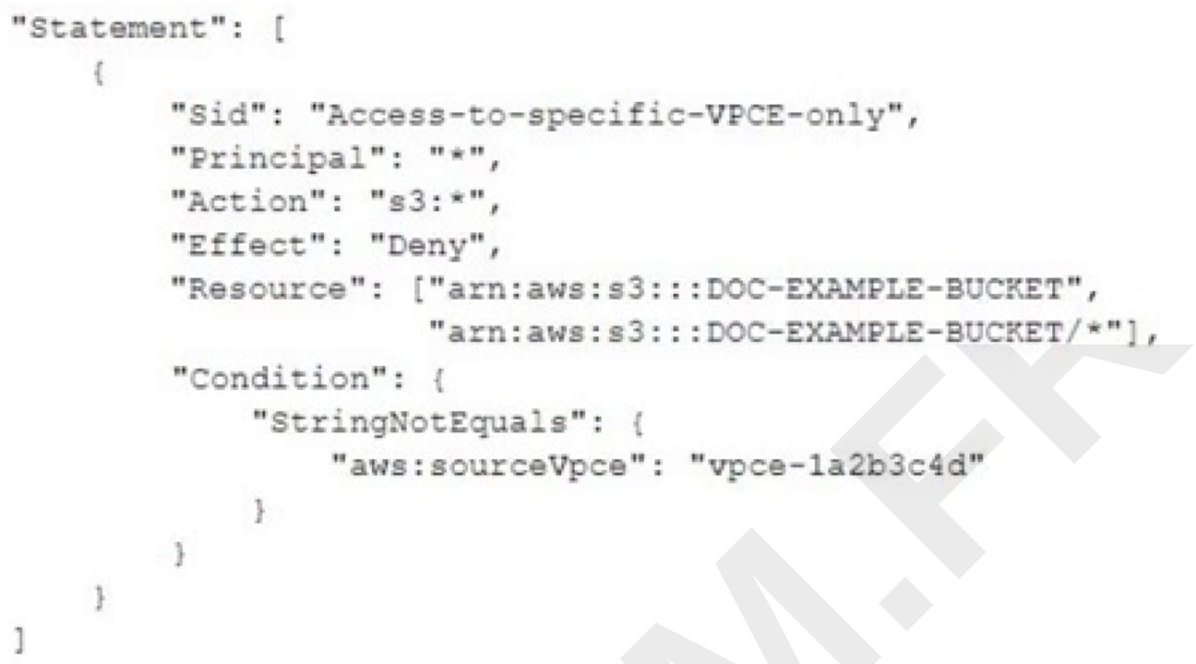
A security engineer needs to configure an Amazon S3 bucket policy to restrict access to an S3 bucket that is named DOC-EXAMPLE-BUCKET. The policy must allow access to only DOC-EXAMPLE-BUCKET from only the following endpoint: vpce-1a2b3c4d. The policy must deny all access to DOC-EXAMPLE-BUCKET if the specified endpoint is not used.

Which bucket policy statement meets these requirements?



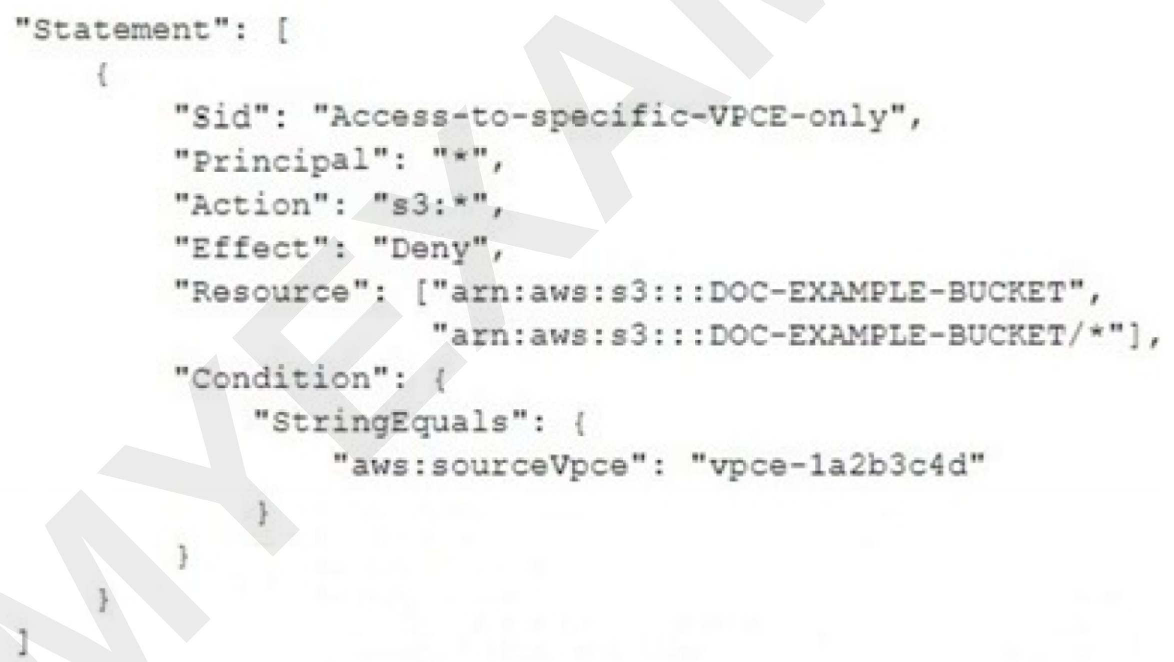
```
"Statement": [
    {
        "Sid": "Access-to-specific-VPCE-only",
        "Principal": "*",
        "Action": "s3:*",
        "Effect": "Allow",
        "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                     "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"],
        "Condition": {
            "StringNotEquals": {
                "aws:sourceVpce": "vpce-1a2b3c4d"
            }
        }
    }
]
```

A.



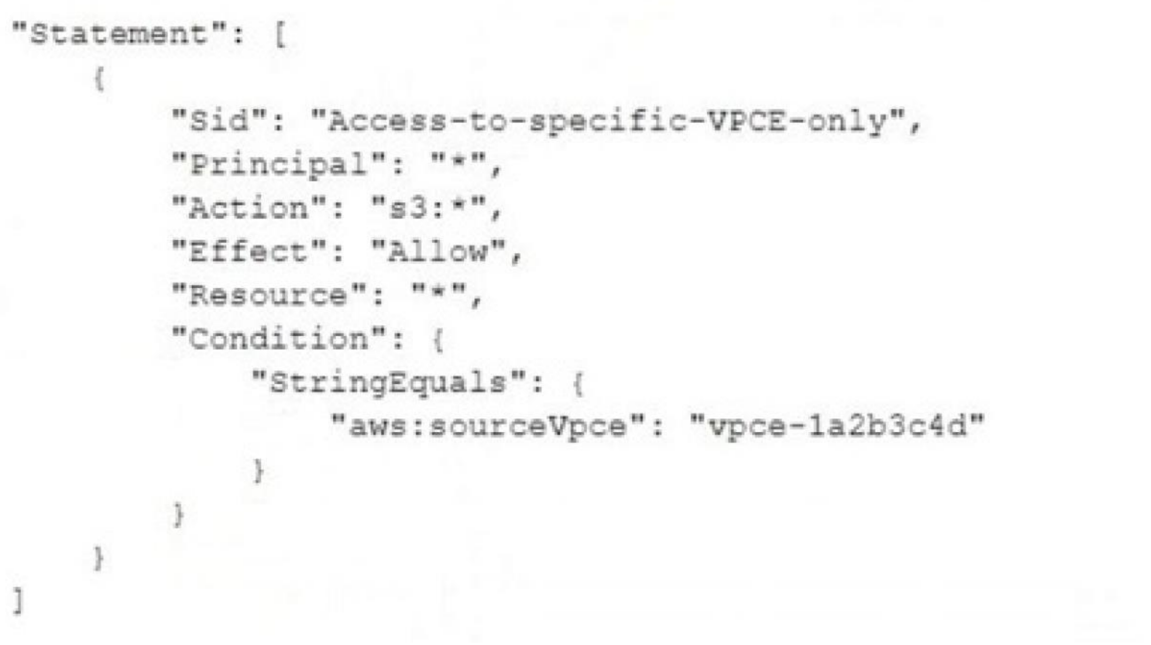
```
"Statement": [
    {
        "Sid": "Access-to-specific-VPCE-only",
        "Principal": "*",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                     "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"],
        "Condition": {
            "StringNotEquals": {
                "aws:sourceVpce": "vpce-1a2b3c4d"
            }
        }
    }
]
```

B.



```
"Statement": [
    {
        "Sid": "Access-to-specific-VPCE-only",
        "Principal": "*",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                     "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"],
        "Condition": {
            "StringEquals": {
                "aws:sourceVpce": "vpce-1a2b3c4d"
            }
        }
    }
]
```

C.

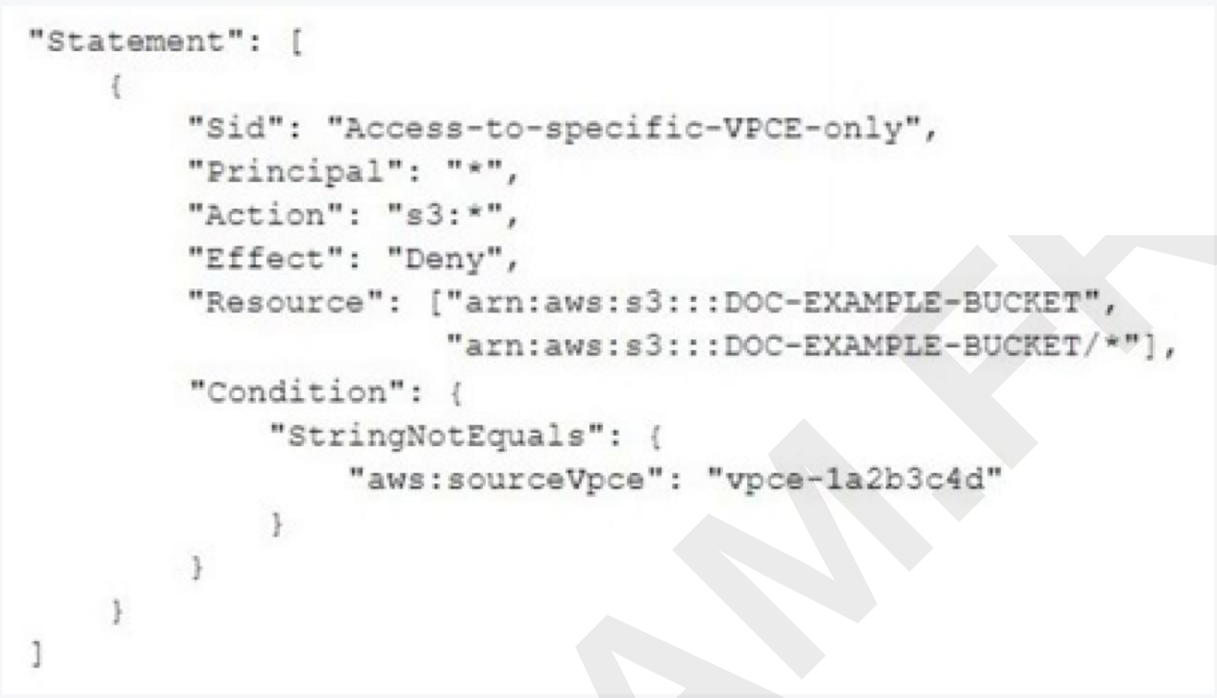


```
"Statement": [
    {
        "Sid": "Access-to-specific-VPCE-only",
        "Principal": "*",
        "Action": "s3:*",
        "Effect": "Allow",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:sourceVpce": "vpce-1a2b3c4d"
            }
        }
    }
]
```

D.

Answer: B

Explanation:



```
"Statement": [
    {
        "Sid": "Access-to-specific-VPCE-only",
        "Principal": "*",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": ["arn:aws:s3:::DOC-EXAMPLE-BUCKET",
                     "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"],
        "Condition": {
            "StringNotEquals": {
                "aws:sourceVpce": "vpce-1a2b3c4d"
            }
        }
    }
]
```

Question: 70

A company has a group of Amazon EC2 instances in a single private subnet of a VPC with no internet gateway attached. A security engineer has installed the Amazon CloudWatch agent on all instances in that subnet to capture logs from a specific application. To ensure that the logs flow securely, the company's networking team has created VPC endpoints for CloudWatch monitoring and CloudWatch logs. The networking team has attached the endpoints to the VPC.

The application is generating logs However, when the security engineer queries CloudWatch, the logs do not appear.

Which combination of steps should the security engineer take to troubleshoot this issue? (Choose three.)

A.Ensure that the EC2 instance profile that is attached to the EC2 instances has permissions to create log streams and write logs.

B.Create a metric filter on the logs so that they can be viewed in the AWS Management Console.

C.Check the CloudWatch agent configuration file on each EC2 instance to make sure that the CloudWatch agent is collecting the proper log files.

D.Check the VPC endpoint policies of both VPC endpoints to ensure that the EC2 instances have permissions to use them.

E.Create a NAT gateway in the subnet so that the EC2 instances can communicate with CloudWatch.

F.Ensure that the security groups allow all the EC2 instances to communicate with each other to aggregate logs before sending.

Answer: ACD

Explanation:

The correct answer is ACD. Here's a breakdown of why each choice is correct:

A. Ensure that the EC2 instance profile that is attached to the EC2 instances has permissions to create log streams and write logs. EC2 instances need the necessary IAM permissions to interact with AWS services, including CloudWatch. Specifically, they must have permission to logs:CreateLogStream and logs:PutLogEvents to create log streams and write log events, respectively. Without these permissions, the CloudWatch agent on the instances will be unable to send logs to CloudWatch, even if the network connectivity is properly configured.
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/create-iam-roles-for-cloudwatch-agent.html

C. Check the CloudWatch agent configuration file on each EC2 instance to make sure that the CloudWatch agent is collecting the proper log files. The CloudWatch agent relies on a configuration file to determine which log files to monitor and where to send them. If the configuration is incorrect (e.g., incorrect file paths, incorrect log group name), the agent might not be collecting the intended logs, or it might be sending logs to a non-existent log group. Inspecting the configuration file is essential for verifying that the agent is properly set up to collect the necessary logs from the correct location on the instances.
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/CloudWatch-Agent-Configuration-File-Details.html

D. Check the VPC endpoint policies of both VPC endpoints to ensure that the EC2 instances have permissions to use them. VPC endpoint policies control which IAM principals can access the service through the endpoint. If the endpoint policy is too restrictive, it might prevent the EC2 instances from sending logs to CloudWatch, even though the network connectivity is established. Verify that the endpoint policy allows the EC2 instances (or the IAM roles associated with them) to perform actions like logs:CreateLogStream and logs:PutLogEvents on the relevant CloudWatch resources. The policy should explicitly allow the instances to use the endpoints for CloudWatch logs and CloudWatch monitoring.
https://docs.aws.amazon.com/vpc/latest/privatelink/vpc-endpoints-access.html

Here's why the other options are incorrect:

B. Create a metric filter on the logs so that they can be viewed in the AWS Management Console. Metric filters are used to extract numerical data from logs to create CloudWatch metrics. They do not affect whether logs are being sent or stored; they only process existing logs. The issue is that no logs are appearing, so a metric filter won't solve that.

E. Create a NAT gateway in the subnet so that the EC2 instances can communicate with CloudWatch. The problem explicitly mentions VPC endpoints are in place for CloudWatch monitoring and CloudWatch logs.

VPC endpoints allow private communication with AWS services within the VPC, so a NAT gateway is not needed in this scenario. A NAT gateway is only necessary when instances need to reach public services, which is not the case when using VPC endpoints.

F. Ensure that the security groups allow all the EC2 instances to communicate with each other to

aggregate logs before sending. The scenario says that the CloudWatch agent has been installed on all instances. Therefore, each instance is individually responsible for sending logs. There is no mention of logs aggregating on a specific EC2 instance before being sent to CloudWatch. Security group rules are about network traffic rules between instances, which is not a part of the current problem.

Question: 71

A company uses AWS Signer with all of the company's AWS Lambda functions. A developer recently stopped working for the company. The company wants to ensure that all the code that the developer wrote can no longer be deployed to the Lambda functions.

Which solution will meet this requirement?

A.Revoke all versions of the signing profile assigned to the developer.

B.Examine the developer's IAM roles. Remove all permissions that grant access to Signer.

C.Re-encrypt all source code with a new AWS Key Management Service (AWS KMS) key.

D.Use Amazon CodeGuru to profile all the code that the Lambda functions use.

Answer: A

Explanation:

The correct answer is A: Revoke all versions of the signing profile assigned to the developer. Here's why:

AWS Signer ensures that only authorized code can be deployed to your AWS Lambda functions. It does this by cryptographically signing the code packages. A signing profile, managed within AWS Signer, defines the signing configuration and is associated with an IAM principal (like a user or role).

If a developer leaves the company, you need to prevent them from deploying malicious or unauthorized code using their old signing credentials. Revoking all versions of the signing profile effectively invalidates any code previously signed by that profile. Lambda functions configured to only accept code signed by a valid, non-revoked signing profile will reject any deployments attempted using the revoked profile.

Let's examine why the other options are incorrect:

B. Examine the developer's IAM roles. Remove all permissions that grant access to Signer. While removing the developer's Signer permissions prevents them from creating new signing profiles or modifying existing ones, it doesn't invalidate the code they already signed with their existing profile. The already-signed code remains valid until the signing profile itself is revoked.

C. Re-encrypt all source code with a new AWS Key Management Service (AWS KMS) key. Re-encrypting the source code doesn't affect the digital signature applied by AWS Signer. The signature is separate from the encryption of the source code itself. AWS Signer does not use KMS keys directly for signing unless you configure Signer to use a KMS key as the backend for a signing certificate (which is uncommon for Lambda). Encryption addresses data confidentiality, not code integrity or authorization.

D. Use Amazon CodeGuru to profile all the code that the Lambda functions use. Amazon CodeGuru is a service for code review and application profiling. While it can help identify potential performance bottlenecks and code quality issues, it doesn't address the problem of unauthorized code deployment through a compromised or terminated user's existing signing profile.

Therefore, revoking the signing profile is the direct and effective way to prevent the former developer from deploying code to your Lambda functions. This ensures that only code signed by valid, authorized profiles can be deployed.

Authoritative Links:

AWS Signer Documentation:https://docs.aws.amazon.com/signer/latest/dg/what-is.html **Revoking Signing Profiles:**https://docs.aws.amazon.com/signer/latest/dg/revoke-signature.html

Question: 72

A company plans to use AWS Key Management Service (AWS KMS) to implement an encryption strategy to protect data at rest. The company requires client-side encryption for company projects. The company is currently conducting multiple projects to test the company's use of AWS KMS. These tests have led to a sudden increase in the company's AWS resource consumption. The test projects include applications that issue multiple requests each second to KMS endpoints for encryption activities.

The company needs to develop a solution that does not throttle the company's ability to use AWS KMS. The solution must improve key usage for client-side encryption and must be cost optimized.

Which solution will meet these requirements?

A.Use keyrings with the AWS Encryption SDK. Use each keyring individually or combine keyrings into a multi-keyring. Decrypt the data by using a keyring that has the primary key in the multi-keyring.

B.Use data key caching. Use the local cache that the AWS Encryption SDK provides with a caching cryptographic materials manager.

C.Use KMS key rotation. Use a local cache in the AWS Encryption SDK with a caching cryptographic materials manager.

D.Use keyrings with the AWS Encryption SDK. Use each keyring individually or combine keyrings into a multi-keyring. Use any of the wrapping keys in the multi-keyring to decrypt the data.

Answer: B

Explanation:

The problem describes a company experiencing AWS KMS throttling due to high request rates from client-side encryption operations during testing. The goal is to reduce KMS API calls to avoid throttling while optimizing cost.

Option B, using data key caching with the AWS Encryption SDK's caching cryptographic materials manager (Caching CMM), directly addresses this issue. The AWS Encryption SDK, when integrated with a Caching CMM, allows applications to reuse data keys locally for encryption and decryption operations, significantly reducing the number of calls to AWS KMS. The application fetches data keys from KMS initially, but subsequently, the Caching CMM stores these keys and reuses them for a configurable period or number of operations, thereby minimizing KMS requests. This caching mechanism directly alleviates the throttling problem and improves the overall performance and cost-effectiveness of the encryption strategy. It aligns with the principle of reducing round trips to centralized services and leveraging local resources for efficiency.

Option A is incorrect because while keyrings in the AWS Encryption SDK provide flexibility in key management, they don't inherently reduce the number of KMS API calls. Decryption still requires access to the KMS keys, potentially leading to the same throttling issues.

Option C is incorrect because KMS key rotation, while essential for security hygiene, doesn't directly address the issue of high request rates causing throttling. Rotating keys only changes the underlying key material, not the frequency of encryption/decryption requests. The inclusion of local cache is also incorrect for the same reason as B is correct.

Option D is incorrect because, similar to option A, keyrings alone don't minimize KMS API calls. Although using multiple wrapping keys provides redundancy, decryption still requires contacting KMS to unwrap the relevant keys, potentially contributing to throttling.

Therefore, data key caching with the AWS Encryption SDK's Caching CMM offers the most effective and cost-optimized solution to mitigate KMS throttling during client-side encryption by minimizing the number of requests sent to AWS KMS.

Relevant links:https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/concepts.htmlhttps://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/services.htmlhttps://aws.amazon.com/blogs/developer/reducing-aws-kms-costs-with-the-aws-encryption-sdk/

Question: 73

A security team is working on a solution that will use Amazon EventBridge to monitor new Amazon S3 objects. The solution will monitor for public access and for changes to any S3 bucket policy or setting that result in public access. The security team configures EventBridge to watch for specific API calls that are logged from AWS CloudTrail. EventBridge has an action to send an email notification through Amazon Simple Notification Service (Amazon SNS) to the security team immediately with details of the API call.

Specifically, the security team wants EventBridge to watch for the s3:PutObjectAcl, s3:DeleteBucketPolicy, and s3:PutBucketPolicy API invocation logs from CloudTrail. While developing the solution in a single account, the security team discovers that the s3:PutObjectAcl API call does not invoke an EventBridge event However, the s3:DeleteBucketPolicy API call and the s3:PutBucketPolicy API call do invoke an event.

The security team has enabled CloudTrail for AWS management events with a basic configuration in the AWS Region in which EventBridge is being tested. Verification of the EventBridge event pattern indicates that the pattern is set up correctly. The security team must implement a solution so that the s3:PutObjectAcl API call will invoke an EventBridge event. The solution must not generate false notifications.

Which solution will meet these requirements?

A.Modify the EventBridge event pattern by selecting Amazon S3. Select All Events as the event type.

B.Modify the EventBridge event pattern by selecting Amazon S3. Select Bucket Level Operations as the event type.

C.Enable CloudTrail Insights to identify unusual API activity.

D.Enable CloudTrail to monitor data events for read and write operations to S3 buckets.

Answer: D

Explanation:

The correct answer is **D. Enable CloudTrail to monitor data events for read and write operations to S3 buckets.** Here's a detailed justification:

The problem lies in understanding how CloudTrail logs API calls. CloudTrail distinguishes between management events and data events.

Management Events: These are events that record management operations performed on resources in your AWS account. Examples include creating, deleting, or modifying resources (e.g., creating an S3 bucket or modifying an IAM role). By default, CloudTrail logs management events. s3:DeleteBucketPolicy and s3:PutBucketPolicy are examples of management events, explaining why EventBridge is triggered for these.

Data Events: These are events that record actions taken on the data stored within a resource. For S3, this includes object-level API operations such as GetObject, PutObject, and, crucially, PutObjectAcl. Data events are not logged by default and must be explicitly enabled.

The security team is attempting to monitor s3:PutObjectAcl, which is a data event. Because CloudTrail is only configured for management events, the s3:PutObjectAcl call is not being logged, and therefore no EventBridge event is triggered. To resolve this, data event logging must be enabled for the S3 buckets of interest in CloudTrail.

Option A is incorrect because while selecting "All Events" for Amazon S3 in EventBridge might seem like it would capture everything, EventBridge still relies on CloudTrail logs as its source. If CloudTrail isn't recording the event, EventBridge won't see it, regardless of the event pattern.

Option B is incorrect because "Bucket Level Operations" typically refers to management operations on the bucket itself (like creating or deleting the bucket), not object-level operations like setting ACLs.

Option C, enabling CloudTrail Insights, is for identifying unusual activity. While potentially useful for security monitoring in general, it doesn't directly address the core issue of the s3:PutObjectAcl call not being logged in the first place. CloudTrail Insights is an analysis tool built on top of CloudTrail logs; if the logs don't contain the events, Insights can't analyze them.

Enabling CloudTrail data events ensures that all object-level API calls, including s3:PutObjectAcl, are logged and become available to EventBridge, triggering the desired notification without generating false positives from unrelated operations.

Authoritative Links:

AWS CloudTrail Concepts:https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-concepts.html
Monitoring S3 Object Operations with CloudTrail Data Events:
https://docs.aws.amazon.com/awscloudtrail/latest/userguide/logging-data-events-with-cloudtrail.html

Question: 74

A company uses Amazon GuardDuty. The company's security team wants all High severity findings to automatically generate a ticket in a third-party ticketing system through email integration.

Which solution will meet this requirement?

A.Create a verified identity for the third-party ticketing email system in Amazon Simple Email Service (Amazon SES). Create an Amazon EventBridge rule that includes an event pattern that matches High severity GuardDuty findings. Specify the SES identity as the target for the EventBridge rule.

B.Create an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the third-party ticketing email system to the SNS topic. Create an Amazon EventBridge rule that includes an event pattern that matches High severity GuardDuty findings. Specify the SNS topic as the target for the EventBridge rule.

C.Use the GuardDuty CreateFilter API operation to build a filter in GuardDuty to monitor for High severity findings. Export the results of the filter to an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the third-party ticketing email system to the SNS topic.

D.Use the GuardDuty CreateFilter API operation to build a filter in GuardDuty to monitor for High severity findings. Create an Amazon Simple Notification Service (Amazon SNS) topic. Subscribe the third-party ticketing email system to the SNS topic. Create an Amazon EventBridge rule that includes an event pattern that matches GuardDuty findings that are selected by the filter. Specify the SNS topic as the target for the EventBridge rule.

Answer: B

Explanation:

The correct answer is B because it leverages the appropriate AWS services for event-driven automation to meet the requirement of generating tickets for high-severity GuardDuty findings in a third-party system via email. Here's a detailed breakdown:

Option B utilizes Amazon EventBridge, which is a serverless event bus service, to listen for specific events. In this scenario, it's configured with a rule that specifically targets high-severity GuardDuty findings. EventBridge's event pattern matching ensures only the desired findings trigger the rule.

Amazon Simple Notification Service (SNS) acts as a central hub to distribute the notifications. The SNS topic is subscribed to the third-party ticketing system's email address, effectively routing the notifications as emails.

When EventBridge detects a high-severity GuardDuty finding based on the defined rule, it publishes a message to the designated SNS topic. SNS then sends an email notification to the third-party ticketing system, thus generating a ticket. This approach is cost-effective, loosely coupled, and scalable.

The other options are less efficient or technically incorrect. Option A uses SES directly, which is typically used for application-to-user email, not event-driven notifications to a ticketing system. SES doesn't offer the filtering capabilities that EventBridge provides based on GuardDuty finding severity. Option C proposes using GuardDuty's CreateFilter API and exporting results to SNS. While this can filter findings, EventBridge is the preferred and more flexible service for reacting to GuardDuty events. Option D combines the CreateFilter API with EventBridge, adding unnecessary complexity. EventBridge can directly filter GuardDuty findings without needing a GuardDuty filter. GuardDuty filters are generally used for filtering within the GuardDuty console itself.

Supporting Links:

Amazon EventBridge:https://aws.amazon.com/eventbridge/
Amazon Simple Notification Service (SNS):https://aws.amazon.com/sns/
Amazon GuardDuty:https://aws.amazon.com/guardduty/

Question: 75

A company is using AWS Organizations to implement a multi-account strategy. The company does not have on-premises infrastructure. All workloads run on AWS. The company currently has eight member accounts. The company anticipates that it will have no more than 20 AWS accounts total at any time.

The company issues a new security policy that contains the following requirements:

•No AWS account should use a VPC within the AWS account for workloads.
•The company should use a centrally managed VPC that all AWS accounts can access to launch workloads in subnets.
•No AWS account should be able to modify another AWS account's application resources within the centrally managed VPC.
•The centrally managed VPC should reside in an existing AWS account that is named Ac-count-A within an organization.

The company uses an AWS CloudFormation template to create a VPC that contains multiple subnets in Account-A. This template exports the subnet IDs through the CloudFormation Outputs section.

Which solution will complete the security setup to meet these requirements?

A.Use a CloudFormation template in the member accounts to launch workloads. Configure the template to use the Fn::ImportValue function to obtain the subnet ID values.

B.Use a transit gateway in the VPC within Account-A. Configure the member accounts to use the transit gateway to access the subnets in Account-A to launch workloads.

C.Use AWS Resource Access Manager (AWS RAM) to share Account-A's VPC subnets with the remaining member accounts. Configure the member accounts to use the shared subnets to launch workloads.

D.Create a peering connection between Account-A and the remaining member accounts. Configure the member accounts to use the subnets in Account-A through the VPC peering connection to launch workloads.

Answer: C

Explanation:

The correct answer is **C: Use AWS Resource Access Manager (AWS RAM) to share Account-A's VPC subnets with the remaining member accounts. Configure the member accounts to use the shared subnets**

to launch workloads.

Here's a detailed justification:

The company requires a centrally managed VPC (in Account-A) that all AWS accounts can access, and that no account can modify another account's application resources within that VPC. AWS RAM directly addresses these requirements in a secure and manageable way. AWS RAM enables you to share AWS resources, like VPC subnets, with other AWS accounts or within your AWS organization. This sharing is controlled and auditable, aligning with security best practices.

Option C allows Account-A (the central VPC account) to maintain control over the VPC infrastructure (subnets, security groups, etc.). The other member accounts can then launch workloads into the shared subnets without having the ability to modify the underlying network infrastructure. This meets the requirement of no account being able to modify another account's application resources.

Let's examine why other options are less suitable:

A: Use a CloudFormation template in the member accounts to launch workloads. Configure the template to use the Fn::ImportValue function to obtain the subnet ID values. While this allows member accounts to launch workloads in Account-A's subnets, it does not provide a centralized sharing mechanism. It also doesn't inherently address the requirement that accounts should not modify each other's resources.

B: Use a transit gateway in the VPC within Account-A. Configure the member accounts to use the transit gateway to access the subnets in Account-A to launch workloads. Transit Gateway primarily facilitates connectivity between VPCs and on-premises networks. While it provides a central hub, it's primarily for routing and doesn't directly enforce resource ownership or modification restrictions within the centrally managed VPC.

D: Create a peering connection between Account-A and the remaining member accounts. Configure the member accounts to use the subnets in Account-A through the VPC peering connection to launch workloads. VPC peering creates network connections between VPCs but doesn't offer a managed sharing environment. This approach quickly becomes complex and hard to manage with even a small number of accounts. It doesn't address the central VPC access and resource modification restriction requirements effectively.

Key Advantages of AWS RAM (Option C):

Centralized Control: Account-A retains control of the VPC and subnets.
Secure Sharing: AWS RAM provides a secure and auditable way to share resources.
Granular Permissions: You can specify which resources are shared and with whom.
Organization Integration: AWS RAM integrates seamlessly with AWS Organizations. **Scalability:** It's a manageable solution, especially for a relatively small number of accounts. **Cost-Effective:** AWS RAM itself doesn't incur any costs.

Here are some authoritative links to further research AWS RAM:

AWS Resource Access Manager (RAM):https://aws.amazon.com/ram/
AWS RAM Documentation:https://docs.aws.amazon.com/ram/latest/userguide/what-is.html
Share your VPC subnets using AWS Resource Access Manager:https://aws.amazon.com/blogs/networking-and-content-delivery/share-your-vpc-subnets-using-aws-resource-access-manager/

In conclusion, AWS RAM is the most appropriate solution to meet the requirements of centralized VPC management, resource sharing, and preventing unauthorized modifications in a multi-account AWS Organizations environment.

Question: 76

A company's security team needs to receive a notification whenever an AWS access key has not been rotated in 90 or more days. A security engineer must develop a solution that provides these notifications automatically.

Which solution will meet these requirements with the LEAST amount of effort?

A.Deploy an AWS Config managed rule to run on a periodic basis of 24 hours. Select the access-keys-rotated managed rule, and set the maxAccessKeyAge parameter to 90 days. Create an Amazon EventBridge rule with an event pattern that matches the compliance type of NON_ COMPLIANT from AWS Config for the managed rule. Configure EventBridge to send an Amazon Simple Notification Service (Amazon SNS) notification to the security team.

B.Create a script to export a .csv file from the AWS Trusted Advisor check for IAM access key rotation. Load the script into an AWS Lambda function that will upload the .csv file to an Amazon S3 bucket. Create an Amazon Athena table query that runs when the .csv file is uploaded to the S3 bucket. Publish the results for any keys older than 90 days by using an invocation of an Amazon Simple Notification Service (Amazon SNS) notification to the security team.

C.Create a script to download the IAM credentials report on a periodic basis. Load the script into an AWS Lambda function that will run on a schedule through Amazon EventBridge. Configure the Lambda script to load the report into memory and to filter the report for records in which the key was last rotated at least 90 days ago. If any records are detected, send an Amazon Simple Notification Service (Amazon SNS) notification to the security team.

D.Create an AWS Lambda function that queries the IAM API to list all the users. Iterate through the users by using the ListAccessKeys operation. Verify that the value in the CreateDate field is not at least 90 days old. Send an Amazon Simple Notification Service (Amazon SNS) notification to the security team if the value is at least 90 days old. Create an Amazon EventBridge rule to schedule the Lambda function to run each day.

Answer: A

Explanation:

The correct answer is A because it offers the most straightforward and automated solution with the least amount of effort by leveraging existing AWS services designed for security and compliance monitoring.

Here's a detailed justification:

AWS Config Managed Rule: AWS Config provides pre-built managed rules for common security and compliance checks. The access-keys-rotated rule specifically checks the age of access keys, which directly addresses the requirement. Using a managed rule minimizes the need for custom coding and maintenance.
https://docs.aws.amazon.com/config/latest/developerguide/evaluate-config.html
Periodic Evaluation: Setting the rule to run periodically (24 hours) ensures daily monitoring of access key rotation status.
EventBridge Integration: Amazon EventBridge allows you to create rules that react to changes in your AWS environment. By creating an event pattern that matches the NON_COMPLIANT compliance type from the Config rule, the system automatically triggers a notification when access keys exceed the 90-day rotation period. https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html
SNS Notification: Amazon Simple Notification Service (SNS) provides a simple and scalable way to send notifications. Configuring EventBridge to send an SNS notification to the security team ensures that they are promptly alerted when non-compliant access keys are detected.
https://docs.aws.amazon.com/sns/latest/dg/sns-what-is.html

The other options are less efficient and require more manual configuration and coding:

Option B: Using AWS Trusted Advisor and Athena involves exporting data, storing it in S3, querying it, and then generating notifications. This is more complex and time-consuming compared to using Config.
Option C: Downloading the IAM credentials report and processing it with a Lambda function requires custom code to parse the report and filter for records older than 90 days. Config provides this functionality out-of-the-box.

Option D: Querying the IAM API directly through Lambda also requires custom code to iterate through users and access keys, and to determine the age of each key. This is less efficient and requires more effort than using AWS Config.

Question: 77

A company maintains an open-source application that is hosted on a public GitHub repository. While creating a new commit to the repository, an engineer uploaded their AWS access key and secret access key. The engineer reported the mistake to a manager, and the manager immediately disabled the access key.

The company needs to assess the impact of the exposed access key. A security engineer must recommend a solution that requires the least possible managerial overhead.

Which solution meets these requirements?

A.Analyze an AWS Identity and Access Management (IAM) use report from AWS Trusted Advisor to see when the access key was last used.

B.Analyze Amazon CloudWatch Logs for activity by searching for the access key.

C.Analyze VPC flow logs for activity by searching for the access key.

D.Analyze a credential report in AWS Identity and Access Management (IAM) to see when the access key was last used.

Answer: D

Explanation:

The best solution is **D. Analyze a credential report in AWS Identity and Access Management (IAM) to see when the access key was last used.**

Here's why:

Direct Information: An IAM credential report provides a comprehensive overview of all users and their credentials within the AWS account, including the last time an access key was used. This offers a quick and direct method to determine if the compromised key was active.

IAM Focus: The credential report is specifically designed for managing and auditing IAM credentials. This aligns perfectly with the incident: a compromised access key associated with an IAM user.

Low Overhead: Generating and analyzing a credential report requires minimal managerial overhead. It's a built-in IAM feature that's simple to generate and interpret.

Avoids Log Scouring Complexity: Options A, B, and C require searching through vast amounts of log data, which is time-consuming and complex. CloudWatch Logs and VPC Flow Logs might contain entries related to the access key, but extracting relevant information and determining the impact could be difficult. Trusted Advisor (Option A) primarily offers best practice recommendations and doesn't directly provide granular usage information.

Cost Effective: Generating and analyzing the credential report won't incur substantial extra costs. This is in comparison to using services like AWS CloudTrail, which would involve additional log ingestion and processing fees.

Therefore, the credential report is the most efficient and straightforward method for assessing the impact of the compromised access key with minimal managerial overhead. It provides immediate insight into whether and when the key was used, allowing for a quicker response and mitigation.

Relevant Links:

Using credential reports - IAM

Question: 78

A company plans to create individual child accounts within an existing organization in AWS Organizations for each of its DevOps teams. AWS CloudTrail has been enabled and configured on all accounts to write audit logs to an Amazon S3 bucket in a centralized AWS account. A security engineer needs to ensure that DevOps team members are unable to modify or disable this configuration.

How can the security engineer meet these requirements?

A.Create an IAM policy that prohibits changes to the specific CloudTrail trail and apply the policy to the AWS account root user.

B.Create an S3 bucket policy in the specified destination account for the CloudTrail trail that prohibits configuration changes from the AWS account root user in the source account.

C.Create an SCP that prohibits changes to the specific CloudTrail trail and apply the SCP to the appropriate organizational unit or account in Organizations.

D.Create an IAM policy that prohibits changes to the specific CloudTrail trail and apply the policy to a new IAM group. Have team members use individual IAM accounts that are members of the new IAM group.

Answer: C

Explanation:

The correct answer is **C: Create an SCP that prohibits changes to the specific CloudTrail trail and apply the SCP to the appropriate organizational unit or account in Organizations.**

Here's a detailed justification:

AWS Organizations provides centralized management and governance across multiple AWS accounts. Service Control Policies (SCPs) are a powerful feature within Organizations that allows administrators to define guardrails or boundaries for the actions that can be performed within member accounts. SCPs are applied to organizational units (OUs) or individual accounts and restrict the permissions of IAM users and roles within those entities.

In this scenario, the goal is to prevent DevOps teams from modifying or disabling the CloudTrail configuration, which is crucial for audit logging and security monitoring. An SCP is the appropriate tool because it operates at the organizational level and enforces restrictions regardless of the IAM policies defined within the individual member accounts. This ensures a consistent and centralized control over the CloudTrail configuration.

Option A is incorrect because applying an IAM policy to the root user might not be sufficient. Root users should be tightly controlled, and relying solely on the root user's IAM policy for such crucial restrictions isn't best practice. Additionally, IAM policies in a member account can be overridden by a more permissive SCP in the management account if one isn't already enforced.

Option B is flawed. While an S3 bucket policy can restrict access to the bucket, it doesn't directly prevent modification of the CloudTrail configuration itself. Moreover, it would be incredibly difficult and likely brittle to only restrict the configuration of CloudTrail via a bucket policy. A malicious actor could likely circumvent those defenses, while SCPs protect CloudTrail configurations directly.

Option D is partially correct but not as comprehensive as Option C. Applying an IAM policy to an IAM group prevents members of that group from making changes, but it only applies to the members of that IAM group.
The member accounts could still have other users or roles with sufficient permissions to alter or disable CloudTrail. Also, it requires management of IAM resources within each member account, whereas an SCP is applied centrally from the AWS Organizations management account. SCPs are better suited for enforcing organization-wide guardrails.

By creating an SCP that denies actions related to updating or deleting the specific CloudTrail trail (e.g., cloudtrail:UpdateTrail, cloudtrail:DeleteTrail) and applying it to the DevOps team's OU or individual accounts, the security engineer can effectively prevent them from tampering with the CloudTrail configuration, ensuring consistent and reliable audit logging.

Authoritative links:

AWS Organizations:https://aws.amazon.com/organizations/
Service Control Policies (SCPs):
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scp.html **AWS CloudTrail:**https://aws.amazon.com/cloudtrail/

Question: 79

A company's policy requires that all API keys be encrypted and stored separately from source code in a centralized security account. This security account is managed by the company's security team. However, an audit revealed that an API key is stored with the source code of an AWS Lambda function in an AWS CodeCommit repository in the DevOps account.

How should the security team securely store the API key?

A.Create a CodeCommit repository in the security account using AWS Key Management Service (AWS KMS) for encryption. Require the development team to migrate the Lambda source code to this repository.

B.Store the API key in an Amazon S3 bucket in the security account using server-side encryption with Amazon S3 managed encryption keys (SSE-S3) to encrypt the key. Create a presigned URL for the S3 key, and specify the URL in a Lambda environmental variable in the AWS CloudFormation template. Update the Lambda function code to retrieve the key using the URL and call the API.

C.Create a secret in AWS Secrets Manager in the security account to store the API key using AWS Key Management Service (AWS KMS) for encryption. Grant access to the IAM role used by the Lambda function so that the function can retrieve the key from Secrets Manager and call the API.

D.Create an encrypted environment variable for the Lambda function to store the API key using AWS Key Management Service (AWS KMS) for encryption. Grant access to the IAM role used by the Lambda function so that the function can decrypt the key at runtime.

Answer: C

Explanation:

Here's a detailed justification for why option C is the best approach for securely storing the API key, along with why the other options are less suitable:

Why Option C (Secrets Manager) is the Best:

Centralized Secret Management: AWS Secrets Manager is specifically designed for managing secrets like API keys, passwords, and database credentials in a centralized and secure manner. This aligns perfectly with the company's policy of storing API keys in a security account managed by the security team.

Encryption at Rest: Secrets Manager automatically encrypts secrets at rest using AWS KMS, further protecting the API key's confidentiality. This fulfills the policy requirement for encryption.

Access Control: Secrets Manager allows fine-grained access control through IAM policies. The security team can grant only the Lambda function's IAM role the permission to retrieve the secret, limiting the exposure of the API key.

Auditing and Rotation: Secrets Manager integrates with AWS CloudTrail for auditing secret access and provides features for automatic secret rotation, improving security posture.

Simplified Lambda Integration: Lambda functions can easily retrieve secrets from Secrets Manager using the AWS SDK, making the integration process straightforward and efficient.
Principle of Least Privilege: Access is granted only to the IAM role that needs it.

Why Other Options are Less Suitable:

Option A (CodeCommit in Security Account): This option focuses on where the source code is stored, but doesn't address the fundamental problem of storing the API key within the code itself. Migrating code does not solve the underlying security problem. Also, it shifts the burden of managing the Lambda function's code to the security team, which is not ideal for a DevOps environment.

Option B (S3 with Presigned URL): While S3 provides encryption at rest, storing the API key directly in S3 and using a presigned URL poses several risks. Presigned URLs can be accidentally shared, granting unauthorized access to the API key. Managing and rotating these URLs can also become cumbersome. Storing a URL in a Lambda environment variable is also considered a secret itself and is not recommended.

Option D (Lambda Environment Variable): Although Lambda environment variables can be encrypted using KMS, they are not designed for long-term secret management. They lack features like auditing, rotation, and centralized control that are essential for security. Also, the environment variables are not in the security account.

In summary: Secrets Manager provides a secure, centralized, and manageable way to store and access the API key, aligning with the company's policy and best practices for secret management. The other options are either less secure, more complex to manage, or do not adequately address the underlying security concerns.

Supporting Links:

AWS Secrets Manager:https://aws.amazon.com/secrets-manager/
Secrets Manager Best Practices:https://docs.aws.amazon.com/secretsmanager/latest/userguide/best-practices.html
Using AWS Secrets Manager with AWS Lambda:
https://docs.aws.amazon.com/secretsmanager/latest/userguide/retrieving-secrets_lambda.html

Question: 80

A security engineer is asked to update an AWS CloudTrail log file prefix for an existing trail. When attempting to save the change in the CloudTrail console, the security engineer receives the following error message: "There is a problem with the bucket policy."

What will enable the security engineer to save the change?

A.Create a new trail with the updated log file prefix, and then delete the original trail. Update the existing bucket policy in the Amazon S3 console with the new log file prefix, and then update the log file prefix in the CloudTrail console.

B.Update the existing bucket policy in the Amazon S3 console to allow the security engineer's principal to perform PutBucketPolicy, and then update the log file prefix in the CloudTrail console.

C.Update the existing bucket policy in the Amazon S3 console with the new log file prefix, and then update the log file prefix in the CloudTrail console.

D.Update the existing bucket policy in the Amazon S3 console to allow the security engineer's principal to perform GetBucketPolicy, and then update the log file prefix in the CloudTrail console.

Answer: C

Explanation:

The error "There is a problem with the bucket policy" when updating a CloudTrail log file prefix indicates that the existing Amazon S3 bucket policy associated with the CloudTrail trail does not allow CloudTrail to write logs with the new specified prefix. CloudTrail uses the S3 bucket policy to ensure it has the necessary permissions to deliver logs to the bucket. When the prefix changes, the policy must be updated to reflect this new path.

Option C directly addresses the problem. It suggests updating the bucket policy to include the new log file prefix. This aligns with the requirement that CloudTrail has explicit permission to write to the bucket using the specified prefix. Once the bucket policy is updated, the security engineer should be able to successfully update the log file prefix in the CloudTrail console, as the necessary permissions are now in place.

Option A, creating a new trail, is unnecessary and inefficient. It involves recreating infrastructure which is not needed to simply update a prefix.

Option B focuses on the security engineer's permissions, but the core issue is CloudTrail's access, not the engineer's ability to modify the bucket policy. Allowing the engineer to PutBucketPolicy doesn't guarantee the correct policy will be in place for CloudTrail's operation. It grants unnecessary broad permissions.

Option D, GetBucketPolicy, is also incorrect because it relates to reading the policy, not modifying it to allow CloudTrail to write logs to the new prefix.

Therefore, only Option C accurately identifies the root cause and provides the correct corrective action: updating the S3 bucket policy with the new log file prefix and subsequently updating the prefix in the CloudTrail configuration.

Refer to the official AWS documentation for more details on CloudTrail S3 bucket policies:

https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-s3-bucket-policy.html