Incident Response Playbook Template

Incident Type

Unintended access to an Amazon Simple Storage Service (Amazon S3) bucket

Introduction

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

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

You are responsible for making your own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) references current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. This document is provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Summary

This Playbook

This playbook outlines response steps for unintended access to an Amazon Simple Storage Service (Amazon S3, or S3) bucket. These steps are based on the NIST Computer Security Incident Handling Guide (https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf) (Special Publication 800-61 Revision 2) that can be used to:

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

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

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

Note that some of the incident response steps noted below may incur costs in your AWS account(s) for services used in either preparing for, or responding to incidents. Customizing this runbook and testing it will help you to determine if additional costs will be incurred. You can use AWS Cost Explorer (https://aws.amazon.com/aws-cost-management/aws-cost-explorer/) and look at costs incurred over a particular time frame (such as when running Game Days) to establish what the possible impact might be. In reviewing this playbook, you will find steps that involve processes that you may not have in place today. Proactively preparing for incidents means you need the right resource configurations, tools and services in place that allow you to respond to an incident. The next section will provide a summary of this incident type, and then cover the five steps (parts 1 - 5) for handling unintended access to an S3 bucket.

This Incident Type

By default, all S3 buckets are private and can be accessed only by users that are explicitly granted access. Sometimes, customers will make changes to their bucket configuration to meet their requirements, or the requirements of their application. These changes (such as changes to a bucket policy, for example) may result in unintended access being granted to IAM principals or unauthenticated users. This playbook covers steps that can be used to deal with unintended access.

Incident Response Process

Part 1: Acquire, Preserve, Document Evidence

1. You become aware that there has been a possible unintended data access to an Amazon S3 bucket. This information could come via different means, for example:

    - An internal ticketing system (the sources of the ticket are varied and could include any of the means below)
    - A message from a contractor or third-party service provider
    - From an alert in one of your own monitoring systems either internal or external to AWS. For example, in AWS, this might include an AWS Config managed rule, AWS CloudTrail via Amazon EventBridge or Amazon CloudWatch Events and Amazon Simple Notification Service (Amazon SNS), via Amazon GuardDuty, AWS Security Hub or a similar service
    - From a threat actor (for example, requesting a ransom or they will disclose data)
    - From a security researcher
    - Via an anonymous tip
    - From a public news article in the press, on a blog or in the news

2. At this point, you may not know if the issue is due to a mis-configured bucket, or a set of compromised credentials:

    1. Use AWS Identity and Access Management (IAM) Access Analyzer (https://docs.aws.amazon.com/IAM/latest/UserGuide/what-is-access-analyzer.html) to determine if any S3 buckets may have overly permissive configuration, giving unintended access to unauthorized principals
    2. Use IAM Access Analyzer to determine if any IAM role trust policy provides unintended access to principals in the same or other AWS accounts

3. Review CloudTrail logs (https://docs.aws.amazon.com/awscloudtrail/latest/userguide/get-and-view-cloudtrail-log-files.html) to determine if recent (prior to the date on which the unintended access occurred, if known) changes had been made to bucket configuration, such as the bucket's Block All Public Access settings, object ACLs, or bucket policies. You may choose to do this using a tool such as Amazon Athena (https://docs.aws.amazon.com/athena/latest/ug/cloudtrail-logs.html#tips-for-querying-cloudtrail-logs)
4. Review CloudTrail logs to determine if recent (prior to unintended access date, if known) changes have been made to IAM role trust policies

3. If you are already aware of the S3 bucket(s) involved, Firstly move to **Part 2** to contain the incident. Once that is done, return here and then move on to step 5. If you have not established which bucket(s) are involved, continue to step 4.

4. If you do not know which bucket is involved:

    1. If you do not know which bucket is involved, but do know which data is involved:
        1. Use internal tools that links that data to a bucket, such as checking a Configuration Management Database (CMDB)
        2. Use the S3 ls command (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-s3-commands.html#using-s3-commands-listing-buckets) from the AWS Command Line Interface (AWS CLI) (https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html) to list out the contents of your S3 bucket(s). For buckets with many objects, you can filter by specifying specific key prefixes in the ls command. This will be especially useful if you already know the data objects involved.
        3. There are several common third-party tools that can also be used to search though files in Amazon S3
        4. Query CloudTrail Data Event logs for one of the files in question, and determine the name of the bucket from the CloudTrail event. You can do this using Amazon Athena (https://docs.aws.amazon.com/athena/latest/ug/cloudtrail-logs.html#query-examples-cloudtrail-logs) This will also provide you the identity of the principal making the API call
    2. If you do not know specifically which bucket or which data was involved, there are several options:
        1. If you know the set of credentials involved, search CloudTrail **Data Events** using a tool such as Amazon Athena, filtering on that set of credentials and s3.amazonaws.com (http://s3.amazonaws.com/) as the event source. If you do not have an Amazon Athena table already configured, you can quickly set one up from the AWS CloudTrail console (https://docs.aws.amazon.com/athena/latest/ug/cloudtrail-logs.html#create-cloudtrail-table-ct).
        2. Review CloudTrail Insights (https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-insights-events.html) (this must be enabled prior to the event) to determine if any insights have been generated that may relate to the time and date of the unintended access, and determine which resources were involved by drilling down into the events highlighted by the specific insight. This will also provide the user identity of the principal that accessed the object(s).
        3. Simply go to the Amazon S3 console and review the "**Access**" column in the table listing all S3 buckets (under the **Buckets** menu). If the column lists a value of **Public** for a bucket, this means that unauthenticated users have access to objects in the bucket, by virtue of object ACLs or the bucket's bucket policy. This is a useful approach for a single account.
        4. If you want to check across multiple accounts in an AWS Organization, you can use Amazon Macie. Go to the Macie console and select **Summary** from the navigation menu. Review the percentage of buckets that are publicly accessible under the **Public** heading of the first table. Next, review the

number of buckets that are shared with other AWS accounts by reviewing the information under the **Shared** heading of the first table.

5. If the buckets are under your administrative control (either in an AWS account you control, or in an AWS account for which you have authorization and appropriate access, for example via an IAM Role), move ahead to **Part 2** to contain the incident, then return to this point and move on to step 7. If they are not under your administrative control, move on to step 6

6. If you do not have administrative control of the S3 buckets involved, you will need to establish a communication channel with the provider/team/individual that does. This may involve opening a new case on the provider's service platform, contacting their customer service, directly contacting your technical point of contact (PoC) in that organization, or something similar. Do this as a matter of urgency, as in Part 2, below, you will need to request that they take action to revoke the changes giving access.

7. Confirm a ticket/case has been raised for the incident. If not, manually raise one.

8. Determine if any user cases are already open for the bucket(s) that can be correlated to any notifications or data you already have in relation to the incident. Document any noted end-user impact/experience of the issue in the relevant ticket. These may include (but not be limited to):

    1. Files missing from the bucket
    2. Unfamiliar new files in the bucket
    3. Files that have had their access settings (such as ACLs) changed, in particular where those files have been made public
    4. Modified bucket permissions, ACLs, Block Public Access settings (all less likely to be noted by regular users, but possible)

9. In the case of automatically created tickets/cases, determine what internal alarms/metrics are currently indicating an issue (what caused the ticket to be created? - see 1a, above)

10. Determine the application impacted (if any, this may be done quickly via Resource Tags, or by using your CMDB)

11. Determine if there are any known events that could be causing service disruption (for example, an application change roll-out that is now resulting in the mishandling API interactions with the S3 bucket.) This could include but may not be limited to:

    1. Writing incorrect files to the bucket
    2. Incorrectly deleting files from the bucket
    3. Modifying file or bucket permissions, ACLs, or Block Public Access settings

12. Determine for the bucket and/or data in question, what is the data classification (https://d1.awsstatic.com/whitepapers/compliance/AWS_Data_Classification.pdf) of the data that has been (or allegedly has been) accessed? The classification level may determine the incident response path taken, however the remainder of this playbook will assume that the classification of the data indicates some level of business impact (whether reputational, financial or other). This classification may be recorded in your CMDB or a specific data classification document.

13. Incident Communications:

1. Identify stakeholder roles from the application entry in the Configuration Management Database (CMDB) entry for that application, or via the application's risk register
2. Open a conference bridge war room for the incident
3. Notify other applicable internal stakeholders including legal personnel, technical teams and developers as required, and add them to the ticket and the war room, so they are updated as the ticket is updated

14. External Communications (for the relevant team/people, not necessarily the first responder - most likely legal, PR, Technical/Account Managers, C-levels, etc.):

    1. Identify customers that could have/are impacted by the incident
    2. Identify customer data that could have been accessed as a result of the issue
    3. Compile this information and *securely* distribute it to the relevant people (for example, using a corporate file share where permissions and access can be controlled)
    4. Ensure your organizations legal counsel is informed and is included in status updates to internal stakeholders and especially in regards to external communications.
    5. If there are regulations in your jurisdiction requiring reporting of such incidents, ensure the people in your organization responsible for notifying local or federal law enforcement agencies are also notified of the event. Consult your legal advisor and/or law enforcement for guidance on collecting and preserving the evidence and chain of custody.

    Those teams/individuals will also likely identify public domain material (news articles, blog posts, tweets, etc.) that mention the incident and determine if/how to respond to these (some factors to consider include: Are the materials factually correct? How much information can be shared and at what time so as not to compromise any ongoing or future investigations? Are there any potential legal ramifications that need to be considered when responding? etc.). This is not a task for the first responder and should not distract that person from the immediate task of handling the technical aspects of the incident.

    6. There may not be regulations, but either open databases, government agencies or NGOs may track this type of activity. Your reporting may assist others.

Part 2: Contain the Incident

By now, ideally, you should have confirmed whether there has been an unintended access incident and whether the incident relates to a set of leaked credentials, misconfigured bucket policies, modified object ACLs, or a combination of these. Firstly, if compromised credentials were involved, disable those credentials or revoke permissions associated with those credentials, thereby preventing any further API activity using the compromised credentials. Next, move on to bucket policies (if relevant). Finally, move on to Block Public Access settings.

1. For the **compromised credential** details obtained from reviewing CloudTrail logs in Part 1, disable those credentials if they are from AWS accounts under your administrative control
    1. If they are long term IAM user credentials, disable them using the IAM console or API (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_access-keys.html)
    2. If they are short term credentials obtained via AWS Security Token Service (AWS STS) they will be associated with an IAM role. There are a couple of options available to disable these:
        1. Revoke all current roles sessions (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_revoke-sessions.html) (note that if the actor has the ability to obtain new credentials, this won't solve the problem)

2. If the actor is able to obtain another set of credentials and the activity continues, it will then be necessary to remove all IAM policies attached to the role, modify the attached policies to block all access, or modify the role's trust policy to prevent the actor from assuming the role. As the credentials will remain valid for the specified time duration once issued, it is important to note that **only** modifying the trust policy will allow any current valid credentials to continue to be used whilst still valid **Note that the above actions will stop all users from using credentials obtained by assuming the role, including any legitimate users or applications**
3. If the role is attached to an EC2 instance, you will also need to review credentials related to the instance (for example, private/public key pairs used for SSH) to ensure that these have not also been compromised, allowing an actor to access an S3 bucket using the role credentials attached to the instance. Two key vectors:
    1. A user obtains access to an EC2 instance using a set of credentials (private key) that allows them to SSH to the instance
    2. The application on the EC2 instance is not correctly configured, and the instance/application form a reverse proxy, or behind certain types of web application firewalls, or has a role that is directly or indirectly internet-facing and a malicious user performs a Server-side Request Forgery (SSRF) with the goal of having the instance's application simply repeat the request to S3 and return the output. Correct application configuration and EC2 features such as IMDSv2 (https://aws.amazon.com/blogs/security/defense-in-depth-open-firewalls-reverse-proxies-ssrf-vulnerabilities-ec2-instance-metadata-service/) can help mitigate these risks

2. The compromised credentials should now be disabled. Verify this by checking the CloudTrail console for the next 30 minutes or so for ongoing credential use, whether by access key, IAM user, or Role.
3. If the credentials used to obtain data are not under your administrative control, and the access was obtained using credentials (i.e., **not** unauthenticated access) the **bucket policy** is too permissive. IAM roles or users from other AWS accounts also need permissions to access a bucket in your account via the bucket policy. Therefore, it is necessary to modify the bucket's bucket policy to restrict access to those principals:
    1. From the bucket(s) identified in Part 1, review the bucket policies to determine which principals have access to the buckets for data plane and/or management plane operations:
        1. **Do this first**: You will need to modify control plane access (for example, S3 PutBucketPolicy) to prevent unauthorized users from making changes to the bucket's configuration (bucket policies, bucket ACL, bucket or account level Block Public Access settings)
        2. You will need to modify data plane access Put*/Get* to prevent unauthorized users from accessing or modifying data that they should not have access to, for all applicable read and write operations
        3. For specific objects of concern, identify any AWS accounts or predefined groups listed as having permission in the object's ACL (https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html) (IAM users cannot be listed individually in an ACL). Another approach (if your permission model does not involve ACLs) is to modify the ACLs for all objects in the bucket to remove all other accounts or predefined groups and simply apply FULL_CONTROL to the bucket owner only (not the object owner - potentially from a different AWS account)
4. If the issue is that the bucket allows public (unauthenticated) access to objects, either via ACLs, the bucket policy ("principal": "*") or a combination thereof, such settings will need to be modified:
    1. the quickest way to remove this access is to modify the bucket's **Block All Public Access** configuration. However note this will impact **all** objects in the bucket, not just the one(s) you are concerned with
    2. For each bucket, in the Amazon S3 console, go to the object, select it and then click on **Permissions >> Access Control List**. There will be a note on the Access Control List button indicating public access. Check

the **Everyone** radio button under the **Public access** row and then remove the checks in the boxes in the ACL's settings. Finally, click **Save. **Note that this doesn't modify an individual existing object's ACL

3. For each object, in the Amazon S3 console, go to the object, select it and then click on **Permissions**. Under the **Public access** row, select the **Everyone** radio button and uncheck all the boxes in the resulting object ACL properties box and click **Save**. If many objects are involved, it will be quicker to use either the AWS CLI or one of the AWS SDKs (https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObjectAcl.html).

Part 3: Eradicate the Incident

Key components of the eradication process include the following:

1. Mitigate any vulnerabilities that were exploited
2. Rotate compromised passwords/credentials

Most of the steps in this section are taken from the security best practices for Amazon S3 (https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-practices.html). Check back with that document frequently for updates.

1. Eradicate any attack vectors related to systems writing to or reading from the S3 bucket(s). As per **Part 2**, if an actor has gained access to systems that are writing to a bucket, the actor could now effectively use the instance's credentials (via the AWS CLI or AWS SDK) on an application instance, for example, to read data from the bucket:

    1. Configure IMDSv2 (https://aws.amazon.com/blogs/security/defense-in-depth-open-firewalls-reverse-proxies-ssrf-vulnerabilities-ec2-instance-metadata-service/) on all application instances/role holding instances

    2. Rotate SSH credentials (https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html) for EC2 Linux. Consider the following steps:

    3. Determine which instances are using that key pair: `aws ec2 describe-instances --query "Reservations[*].Instances[].[KeyName, InstanceId]"` (Windows CMD example)

    4. Decide on the following:

        - If the instances should be terminated
            1. Establish if the instances can be terminated without impacting your application (for example, using your CMDB, your EC2 instance tags, etc.)
        - Is it sufficient to rotate an instance's user credentials by:
            - Modifying the contents of the `authorized_keys` file; or,
            - Removing the user; and/or,
            - Remove/modify the user's .ssh directory
            - On Windows, modify the allowed RDP users via `Remote Desktop Users`.

        At scale, use AWS Systems Manager to manage instance configuration (https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-ssm-docs.html).

    5. If the instances have been launched using some kind of scaling mechanism:

6. Determine if you have any Launch Configurations (https://docs.aws.amazon.com/autoscaling/ec2/userguide/LaunchConfiguration.html) referencing the key pair. If so, create either a new Launch Configuration referencing the new key pair and, associate it to the required resources, such as EC2 Auto Scaling Groups, or instead move on to step 4 and create a launch Template, giving you all the current features for EC2 Auto Scaling

7. Determine if you have any existing Launch Templates (https://docs.aws.amazon.com/autoscaling/ec2/userguide/LaunchTemplates.html) for EC2 referencing the existing key pair. If so, delete that template, follow the steps below and then create a new template once a new key pair has been created.

8. Delete the key pair from your EC2 console

9. Terminate the instances

10. Create a new key pair in AWS and issue it to any required personnel

11. Launch instances using the new key pair

12. Update user passwords for domain or local users using Microsoft Windows Remote Desktop Protocol (RDP) and verify users in the Remote Desktop Users group within Windows and/or Group Policy (for example, if you are adding Active Directory Global Security Groups to the Remote Desktop Users group on local operating systems, a fairly common Group Policy task)

13. Revoke credentials for all existing role sessions (https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_revoke-sessions.html). This will block access to AWS APIs from the EC2 instance until that instance obtains a new set of role credentials. The instance can obtain new role credentials by waiting until the existing set of credentials are rotated automatically, or by detaching and re-attaching the same role to the EC2 instance. If there are concerns that a threat actor has remote access to the instance (for example, by SSH or RDP), per step 2 in this section, terminate the existing EC2 instance and create a new one after revoking existing sessions. Note that if the instance is accessed using Key Pairs defined in EC2 (https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) you should determine if other instances are using the same key pairs and take appropriate action to prevent the threat actor also accessing those instances. Such steps might include:

2. Mitigate vulnerabilities related to your service configurations. For example, with S3, determine which configuration items need to be updated:

    1. Return modified object ACLs to their secure settings, removing unauthenticated access permissions
    2. Implement least privilege in bucket policies
    3. Ensure only appropriate principals have permission to perform data or control plane operations on the bucket
    4. Ensure IAM policies are appropriately scoped and if necessary, Permission Boundaries (https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_boundaries.html) and/or Service Control Policies (https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps_examples.html) have been deployed appropriately
    5. Encrypt data at rest using Amazon S3 Server Side Encryption (https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html) - this protects the data at rest,

and means that principals must have permissions in **both **S3 and AWS Key Management Service (AWS KMS) (the AWS KMS Customer Master Key (CMK) key policy) to access the data

6. Enable versioning (https://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html) in S3 to enable data recovery, and consider:
7. Object Lock (https://docs.aws.amazon.com/AmazonS3/latest/dev/object-lock.html) to prevent the deletion of critical data

3. If credential compromise was involved:

    1. Flag for review the process for handling user credentials during Move/Add/Change (MAC) procedures for the post-incident activity
    2. The compromised credentials should have been disabled in Part 2, you may now consider issuing new credentials to impacted users
    3. Close down/remove user accounts or roles if they are not valid accounts/roles (or IAM IdPs, AWS Single Sign-On (AWS SSO) instances, etc.)
    4. Enable MFA Delete for S3 (https://docs.aws.amazon.com/AmazonS3/latest/userguide/DeletingObjects.html#DeletingObjectsfromanMFA-EnabledBucket), which means users must provide a valid MFA token along with the request to delete any versioned object in the bucket. This requires also enabling S3 Versioning
    5. If you have objects that need to be public, rather than setting the ACL so that the object is world-readable, consider using an S3 pre-signed URL with a suitable expiry time and have your application pass that to the unauthenticated user to allow them to access the object.

Part 4: Recover from the Incident

The next step is to determine if the incident has been mitigated, or if additional tuning is needed to eradicate it. This includes reviewing pre- and post-mitigation logs to:

- Determine the impact of the mitigations already performed
- Identify any other attack signatures to attempt to eradicate or further mitigate the incident

Once this has been completed, it will likely be necessary to restore any lost or corrupted data.

Restore lost or modified data:

1. If you have S3 Versioning (https://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html), restore an earlier version of the object (https://docs.aws.amazon.com/AmazonS3/latest/dev/RestoringPreviousVersions.html) to replace the modified object. If the object has been deleted using a simple delete command (without specifying a version ID for the object, it will now have a delete marker (https://docs.aws.amazon.com/AmazonS3/latest/dev/DeleteMarker.html) in place of the object. you can delete the delete marker (https://docs.aws.amazon.com/AmazonS3/latest/dev/RemDelMarker.html) to recover the object
2. If the object was deleted and the delete command specified an object Version ID, that version of the object will be permanently deleted. You will need to establish if previous versions of the object still exist, or of the object can be retrieved from another location (see below steps)
3. If you move data to other S3 storage classes (S3 IA, Amazon Simple Storage Service Glacier, etc.) as part of a back-up or archival process, determine if the files still exist in that storage class and retrieve them (remember that files *can be deleted* from other storage classes). Lifecycle policies

(https://docs.aws.amazon.com/AmazonS3/latest/dev/lifecycle-transition-general-considerations.html) are used to move objects between storage classes
4. If you are using S3 Cross Region Replication (CRR) restore the data from the target bucket to the source bucket as necessary
5. If none of the previous actions allow you to restore the deleted or modified data, you will need to determine if the data is retrievable from another source (such as an on-premise system or an S3 bucket in a different account

From the logs obtained in PART 1, review those obtained at the time of the initial investigation. Now that mitigations are in place, you need to go back and obtain those metrics and logs again, and compare them to the incident metrics and logs obtained earlier.

If data plane activity has returned to pre-attack levels and the logs show no further evidence of malicious use, the attack has been mitigated (at least for now). Continue to monitor the service post-attack; if suspicious data plane activity reoccurs, take the following steps:

1. Return to Part 1 and follow those steps again
2. Review logs and related data to determine if you have correctly identified the attack vector, or if the actor has changed the vector in response to the mitigation

Part 5: Post-Incident Activity

This “sharpen the saw” activity helps teams to assess their response to the actual incident, determine what worked and what didn’t, update the process based on that information and record these findings.

1. Review how the incident was handled, as well as the incident handling process generally, with key stakeholders identified in Part 1.
2. Document lessons learned, including attack vector(s) mitigation(s), misconfiguration, etc.
3. Store the artifacts from this process with the application information in the CMDB entry for the application and also in the CMDB entry for the S3 bucket and the data leak incident response process.
4. Update risk documents based on any newly discovered threat/vulnerability combinations that were discovered as a result of lessons learned
5. If new application or infrastructure configuration is required to mitigate any newly identified risks, conduct these change activities and update application and component configuration in the CMDB
6. Ask the following questions and assign people to resolve any issues that come up as a result:
    1. What information would have helped me respond to this incident more swiftly?
    2. What detection would have alerted me to the issue sooner?
    3. What configuration (technical or process) would have prevented this data being exposed in this way?
    4. What automation, tooling or access would have made it easier to investigate the root cause?
7. For any actions that are raised as the result of parts 3, 4 and 5, assign those actions and follow up to ensure they are completed