Version 4.5

August 03, 2018

ASSESSMENT REPORT:
AWS Post-Exploitation Assessment

Contoso

Darin Allison

RHINO SECURITY LABS

ASSESSMENT INFORMATION

RHINO SECURITY LAB DETAILS

Account Executive

Austin Tippett
Account Executive
austin.tippett@rhinosecuritylabs.com
206.408.8009

Assessment Team

Benjamin Caudill (Lead Consultant)
benjamin.caudill@rhinosecuritylabs.com
888.944.8679 x700

Spencer Gietzen
spencer.gietzen@rhinosecuritylabs.com
888.944.8679 x719

Project Manager

David Moratti
david.moratti@rhinosecuritylabs.com
888.944.8679 x704

CLIENT DETAILS

Company Information

Contoso
1234 East Pike St.
Seattle, WA
98122

Contact Information

Darin Allison
Director of Vulnerability Management
darin.allison@contoso.com
555.555.0199

ASSESSMENT SCOPE SUMMARY

Engagement Timeframe

07/24/2018 - 08/03/2018

Engagement Scope

The purpose of this testing is to evaluate and test risk associated with Contoso assets by simulating a real-world attacker.

Project ID: Contoso-AWS-V4.5-04-23-2018

Report Date: August 03, 2018

ENGAGEMENT OVERVIEW

Rhino Security Labs specializes in AWS Post-Exploitation Assessment techniques and practices, identifying areas for improvement. At Rhino Security Labs we specialize in manual assessments that go beyond basic automated tests to identify real attack vectors that can be used against your environment.

With decades of combined experience, Rhino Security Labs is at the forefront of cloud security and penetration testing. With a veteran team of subject matter experts, you can be sure every resource is an authority in their field.

SERVICE DESCRIPTION

Penetration Testing is the process of simulating real-world attacks by using the same techniques as malicious hackers. For a security assessment that goes beyond a simple vulnerability scanner, you need experts in the industry.

EXTENSIVE AWS POST-EXPLOITATION ASSESSMENT

AWS Post-Exploitation testing assesses the organization's security from the perspective of an attacker who has just gained access to the environment.

AWS security issues are not only growing in numbers with the great number of new services and features constantly being released, they're also growing in complexity.

In addition to testing for vulnerabilities, this assessment tests the organizations detection and response capabilities, confirming the effectiveness of SIEM and log aggregation technologies.

CAMPAIGN OBJECTIVES

VULNERABILITY IDENTIFICATION

Rhino Security Labs' consultants use the results of the automated scan, paired with their expert knowledge and experience, to finally conduct a manual security analysis of the client's environment. Our assessors attempt to exploit and gain remote unauthorized access to data and systems through misconfigurations and exploits. The detailed results of both the vulnerability scan and the manual testing are shown in this report.

YOUR ASSESSMENT TEAM

Passionate and forward-thinking, our consultants bring decades of combined technical experience as top-tier researchers, penetration testers, application security experts, and more. Drawing from security experience in the US military, leading technology firms, defense contractors, and Fortune 50 companies, we pride ourselves on both depth and breadth of information.

David Moratti - *Technical Project Manager*

David brings a breadth of information security education and experience. David manages all Rhino Security Labs engagements, performing the penetration testing for many of the engagements himself. With a degree in information security at the University of Washington, David is uniquely able to speak to both technologists and management in language understood and applied by both parties.

Benjamin Caudill - *CEO & Engagement Manager*

Benjamin Caudill is an adept cybersecurity professional, researcher, and entrepreneur. A veteran of the defense and aerospace industry, Mr. Caudill led investigations into advanced cyber-attacks, coordinating with federal intelligence communities on complex engagements. As Founder and CEO of Rhino Security Labs, Mr. Caudill has built the boutique security firm and turned it into a major player in the penetration testing market. In addition to his executive role, Mr. Caudill oversees company research and development, ensuring the continued development of key offensive technologies.

Spencer Gietzen - *Penetration Tester*

Spencer Gietzen came into the cyber security field with a background in web and software development. His primary focus as a penetration tester is security relating to Amazon Web Services, mobile applications, and web applications, where he has found success in discovering multiple vulnerabilities and attack vectors through extensive research. Spencer is also the lead developer of our internal AWS post exploitation framework, Pacu.

PROCESS AND METHODOLOGY

Rhino Security Labs used a comprehensive methodology to provide a security review of Contoso's AWS environment. This process begins with detailed scanning and research into the architecture and environment, with the performance of automated testing for known vulnerabilities. Manual exploitation of vulnerabilities follows, for the purpose of detecting and exploiting security weaknesses in the environment.

1

Reconnaissance

The primary goal in this process is to discover crucial data about Contoso's environment, providing the foundation for a tailored penetration test. Reconnaissance is carried out via automated scanners, as well as manual fingerprinting and discovery.

2

Automated Testing

Rhino Security Labs used a vulnerability scanner to provide a foundation for the full manual assessment, and each finding is manually verified to ensure accuracy and remove false positives.

3

Exploration and Verification

Rhino Security Labs' consultants use the results of the automated scan, paired with their expert knowledge and experience, to finally conduct a manual security analysis of the client's environment. The detailed results of both the vulnerability scan and the manual testing are shown in the tables below.

4

Assessment Reporting

Once the engagement is complete, Rhino Security Labs delivers a detailed analysis and threat report, including remediation steps. Our consultants set an industry standard for clear and concise reports, prioritizing the highest risk vulnerabilities first. The assessment includes the following:

- Executive Summary
- Strategic Strengths and Weaknesses
- Identified Vulnerabilities and Risk Ratings
- Detailed Risk Remediation Steps
- Assets and Data Compromised During Assessment

5

Optional Remediation

As an optional addition to the standard assessment, Rhino Security Labs provides remediation retesting for all vulnerabilities listed in the report. At the conclusion of the remediation testing and request of the client, Rhino Security Labs will update the report with a new risk level determination and mark which vulnerabilities in the report were in fact remediated to warrant a new risk level.

SCOPING AND RULES OF ENGAGEMENT

While real attackers have no limits on AWS Post-Exploitation Assessments, we do not engage in penetration testing activities that threaten our ethics and personal privacy.

Constraints

The following limitations were placed upon this engagement, as agreed upon with Contoso:

- No shutting down EC2 instances.
- No deletion of data
- Start with developer level privileges to simulate compromised credentials.

EXECUTIVE SUMMARY OF FINDINGS

Rhino Security Labs conducted an AWS Post-Exploitation Assessment for Contoso. This test was performed to assess Contoso's defensive posture and provide security assistance through proactively identifying vulnerabilities, validating their severity, and providing remediation steps.

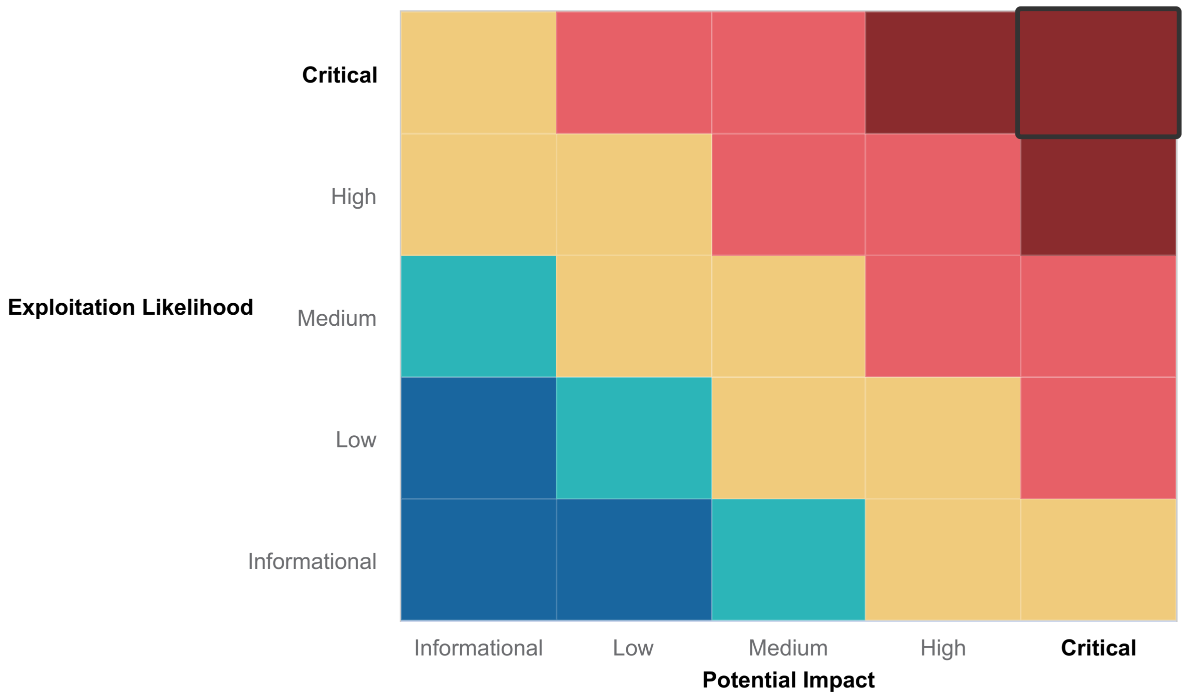
Rhino Security Labs reviewed the security of Contoso's infrastructure and has determined a Critical risk of compromise from external attackers, as shown by the presence of the vulnerabilities detailed in this report.

The detailed findings and remediation recommendations for these assessments may be found later in the report.

AWS Post-Exploitation Risk Rating

Rhino Security Labs calculates AWS post-exploitation risk based on Exploitation Likelihood (ease of exploitation) and Potential Impact (potential business Impact to the environment).

OVERALL RISK RATING: CRITICAL



Summary of Strengths

While Rhino Security Labs was tasked with finding issues and vulnerabilities dealing with the current environment, it is useful to know when positive findings appear. Understanding the strengths of the current environment can reinforce security best practices and provide strategy and direction toward a robust defensive posture. The following traits were identified as strengths in Contoso's environment.

1. Multi-factor authentication required for most API activity.
2. Strict EC2/VPC ingress/egress rules.

Summary of Weaknesses

Rhino Security Labs discovered and investigated many vulnerabilities during the course of its assessments for Contoso. We have categorized these vulnerabilities into general weaknesses across the current environment, and provide direction toward remediation for a more secure enterprise.

1. Many user accounts with the ability to escalate permissions.
2. Many unencrypted EBS volumes and snapshots.

Strategic Recommendations

Not all security weaknesses are technical in nature, nor can they all be remediated by security personnel. Companies often have to focus on the root security issues and resolve them at their core. These strategic steps are changes to the operational policy of the organization. Rhino Security Labs recommends the following strategic steps for improving the company's security.

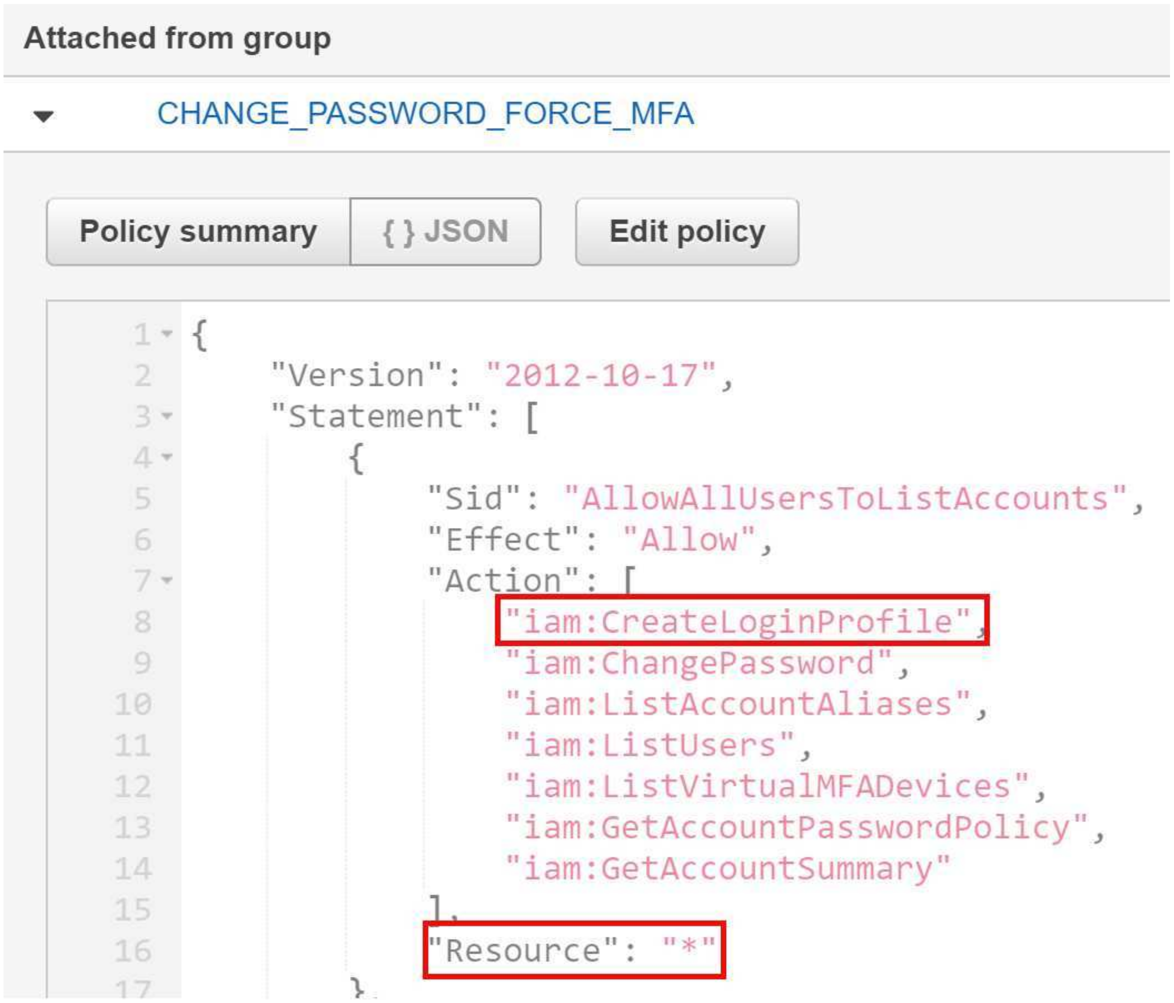
1. Reassess IAM policies to ensure no user is able to escalate their privileges beyond what they are allowed.
2. Always use encryption for different resources in the AWS account to protect sensitive data from being stored in plaintext.

EXECUTIVE SUMMARY NARRATIVE

The assessment began with the assessor receiving a set of credentials that are identical to that of a developer for Contoso. This was to simulate a realistic situation where in some manner the credentials had been compromised by an external attacker or an internal rogue employee.

The assessor began by enumerating the permissions that were available on the IAM account by using the privilege escalation module of an internally built AWS pentesting tool, Pacu. The module attempts to enumerate access of the current account and then attempts to escalate privileges through any available methods.

Using the permissions supplied to the assessor at the beginning of the engagement, they were able to escalate to full administrator access. The user account had the `iam:CreateLoginProfile` permission with its allowed resources set to "*".



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllUsersToListAccounts",
            "Effect": "Allow",
            "Action": [
                "iam:CreateLoginProfile",
                "iam:ChangePassword",
                "iam:ListAccountAliases",
                "iam:ListUsers",
                "iam:ListVirtualMFADevices",
                "iam:GetAccountPasswordPolicy",
                "iam:GetAccountSummary"
            ],
            "Resource": "*"
        },
```

This allowed the assessor to create a login profile for any user in the account and login as them. Two users that had full IAM access and no passwords were found, those are listed here:

- arn:aws:iam::000000000000:user/vault-prod-aws-athena-mount
- arn:aws:iam::000000000000:user/vault-qa-aws-athena-mount

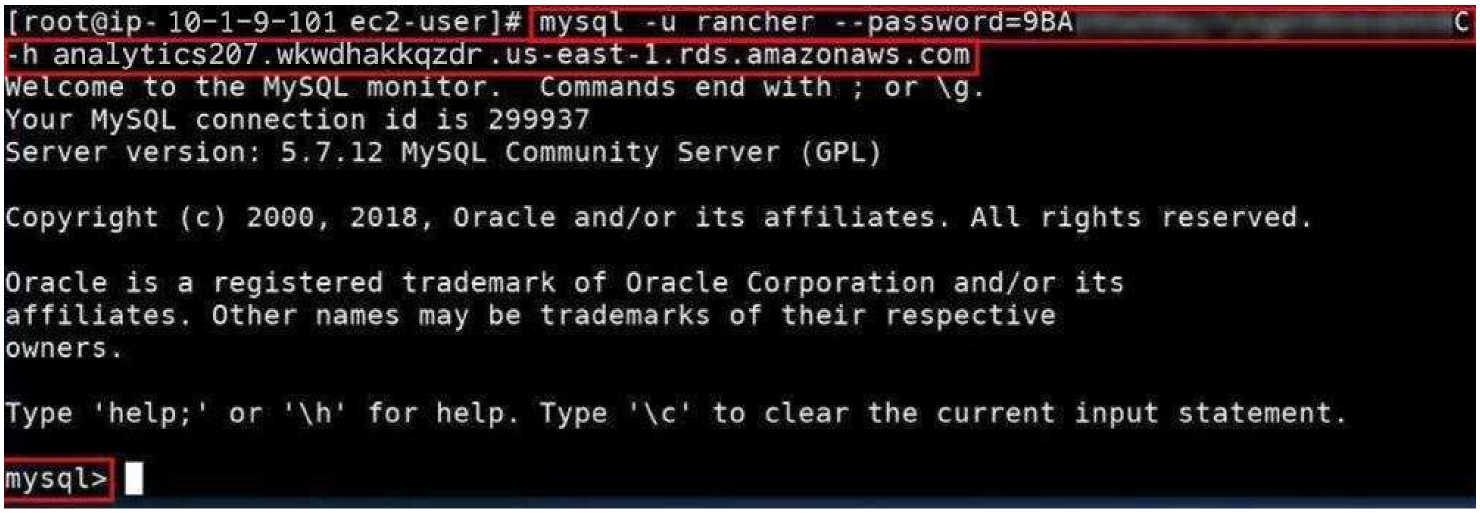
The assessor created a password for and logged in as the vault-ga-aws-athena-mount user, where they then attached an administrator policy to their original user.

With the original account now being elevated to administrator privileges, they began to inspect the S3 Buckets looking for sensitive information that might be useful in expanding access further. The assessor discovered many credentials relating to the web servers and associated services and applications. These sensitive files were located at s3://contoso-devops/analytics207/*, where there were multiple files for each environment setup for the web applications. The assessor was able to find AWS keys, database passwords, and web app credentials contained within these.

The assessor found sensitive data in an S3 bucket in the account. Using the staging-bastion.pem SSH private key that was discovered, the assessor was able to access the EC2 instance "staging-bastion - bastion1" (i-0cba9e1bbbbb120f1f).

With access to that instance, the assessor then began creating snapshots of existing Elastic Block Store Volumes, then creating new volumes from those snapshots. In doing this, the assessor was able to copy and access EBS volumes without ever detaching them from their instances and disrupting service.

Now at this point, the assessor had valid credentials to the analytics207 RDS database, but no way to access the database to try and login due to the RDS and VPC settings. To work around this, the assessor launched an EC2 instance into a public subnet that was located in the same VPC as the analytics207 RDS database. From this EC2 instance, the assessor was able to connect to the database successfully using the "mysql" command line tool. The screenshot below shows the assessor authenticating with the database.

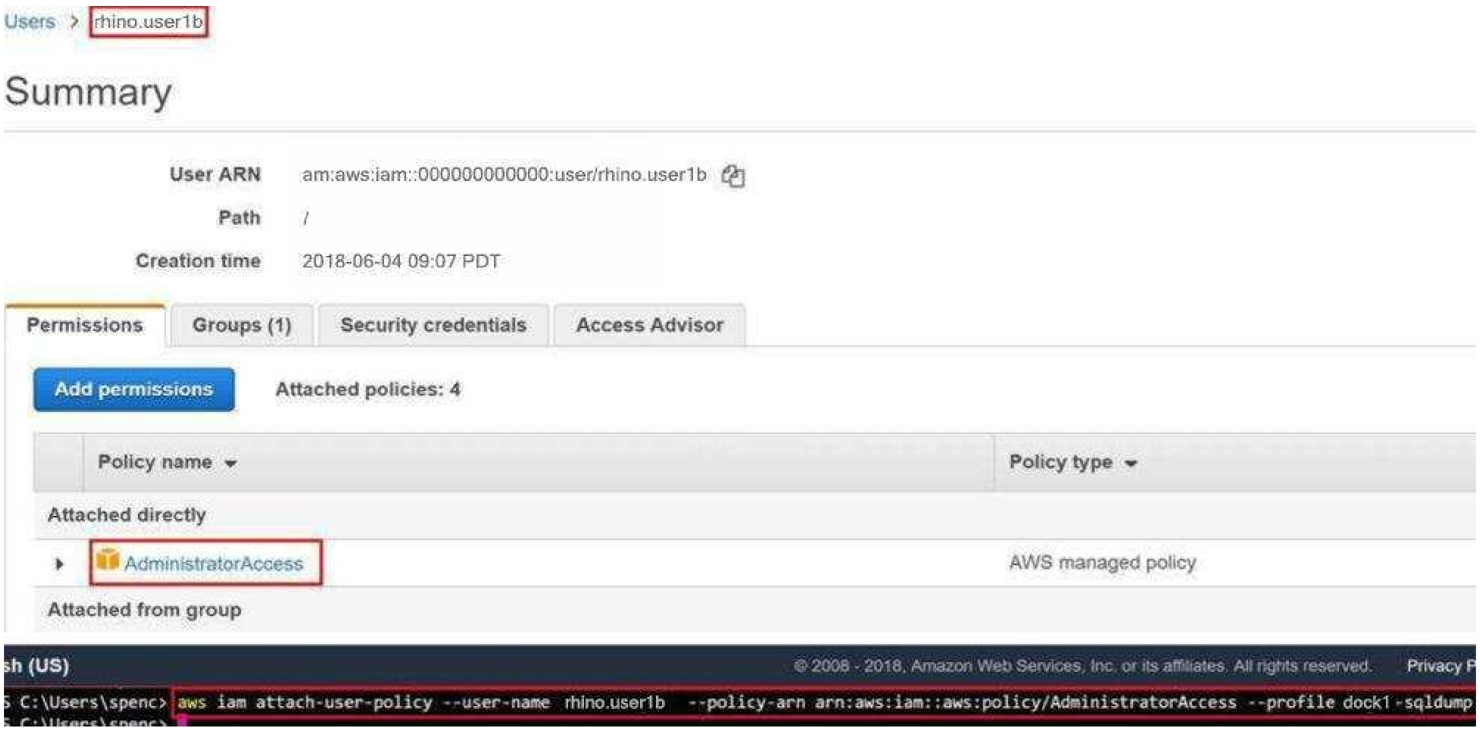


```
[root@ip-10-1-9-101 ec2-user]# mysql -u rancher --password=9BA[redacted]C
-h analytics207.wkwdhakkqzdr.us-east-1.rds.amazonaws.com
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 299937
Server version: 5.7.12 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

Given this access, the assessor used the command line tool "mysqldump" to create a SQL dump of the entire "analytics207"database. They then transferred this file back to their workstation to analyze the data. At this point, they terminated the EC2 instance that was run to access the RDS database.

While analyzing the data, the assessor discovered the MySQL table "audit_logs", which seemed to contain some very sensitive information, including an AWS access key ID and secret access key. By checking the access key ID in AWS IAM, it was discovered that it belonged to the IAM user john.doe(arn:aws:iam::0000000000:user/john.doe). john.doe had administrator access to the AWS environment.

The assessor created an account with the original developer privileges that they were given and recreated this secondary attack chain. This was to test if there was multiple methods to escalate to an AWS administrator. The screenshot below shows the assessor using john.doe's AWS keys to attach the AWS managed Administrator Access policy to their second IAM user and the resulting attached policy in the AWS console. Confirming a secondary route of obtaining administrator credentials.



User ARN: arn:aws:iam::000000000000:user/rhino.user1b
Path: /
Creation time: 2018-06-04 09:07 PDT

Attached policies: 4

Policy name	Policy type
Attached directly	
AdministratorAccess	AWS managed policy
Attached from group	

```
S C:\Users\spenc> aws iam attach-user-policy --user-name rhino.user1b --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --profile dock1-sqldump
S C:\Users\spenc>
```

At this point in the engagement there were two confirmed methods to obtain AWS Administrator access, the assessor had dumped a MySQL database with sensitive information, and had access to S3 Buckets and EC2 instances in the environment. The assessor spent the rest of the engagement looking for misconfigurations and ways that an attacker could establish persistence in the environment.

SUMMARY VULNERABILITY OVERVIEW

Rhino Security Labs performed an AWS Post-Exploitation Assessment for Contoso on 07/24/2018 - 08/03/2018. This assessment utilized both commercial and proprietary tools for the initial mapping and reconnaissance of the site(s), as well as custom tools and scripts for unique vulnerabilities. During the manual analysis, assessors attempted to leverage discovered vulnerabilities and test for key security flaws. The following vulnerabilities were determined to be of highest risk, based on several factors including asset criticality, threat likelihood, and vulnerability severity.

Vulnerability Risk Definition and Criteria

The risk ratings assigned to each vulnerability are determined by averaging several aspects of the exploit and the environment, including reputation, difficulty, and criticality.

CRITICAL	Critical vulnerabilities pose a serious threat to an organization's security, and should be fixed immediately. They may provide a total compromise of the target environment, or similar critical impacts.
HIGH	High risk vulnerabilities provide a serious risk to the company environment and should be corrected promptly. These issues can significantly affect the organization's security posture.
MEDIUM	Medium severity vulnerabilities represent a moderate risk to the environment. They may require additional context before remediation but should be remediated after critical and high risks.
LOW	Low severity vulnerabilities provide minimal risk to the target environment, and often theoretical in nature. Remediation of low risks is often a lower priority than other security hardening techniques.
INFORMATIONAL	Informational vulnerabilities have little-or-no impact to the target scope by themselves. They are included however, as they may be a risk when combined with other circumstances or technologies not currently in place. Remediation of informational items is not necessary.

VULNERABILITY SUMMARY TABLE

The following vulnerabilities were found within each risk level. It is important to know that total vulnerabilities is not a factor in determining risk level. Risk level is depends upon the severity of the vulnerabilities found.

2	4	4	5	1
Critical	High	Medium	Low	Informational

Vulnerability ID - Name And Remediation	Risk Level
C1 - OVERLY PERMISSIVE IAM RESOURCE Follow the principle of least privilege where possible.	CRITICAL
C2 - PRIVILEGE ESCALATION TO AWS ADMINISTRATOR Review the attack chain taken by the assessor and reevaluate the relevant access controls at each pivot point.	CRITICAL
H1 - EC2 USER DATA SENSITIVE INFORMATION LEAKAGE Remove hardcoded secrets and passwords from EC2 User Data.	HIGH
H2 - CLOUDTRAIL LOGGING DISABLED Create a trail that applies to all regions.	HIGH
H3 - AWS S3 BUCKET DATA LEAKAGE Restrict bucket permissions to be accessible only by users or servers who need access to its resources.	HIGH
H4 - WEAK IAM PASSWORD POLICY Implement a password policy that ensures strong, high-entropy passwords for all users.	HIGH
M1 - REDSHIFT CLUSTER DATABASE ENCRYPTION DISABLED Enable encryption for all affected databases.	MEDIUM
M2 - VPC FLOW LOGS DISABLED Enable flow logs on the affected EC2 network interfaces.	MEDIUM
M3 - REDSHIFT PARAMETER GROUP SSL NOT REQUIRED Require SSL for all connections.	MEDIUM
M4 - NO IAM USER ACCESS KEY ROTATION Implement a plan to regularly rotate all IAM user access keys.	MEDIUM

L1 - UNENCRYPTED ELASTIC BLOCK STORE (EBS) SNAPSHOTS — LOW

Enable encryption for all EBS volumes, so that all future EBS snapshots will also be encrypted.

L2 - S3 BUCKET ACCESS LOGGING NOT ENABLED — LOW

Enable access logging for the affected S3 buckets.

L3 - S3 BUCKET VERSIONING NOT ENABLED — LOW

Enable versioning for the affected S3 buckets.

L4 - REDSHIFT USER ACTIVITY LOGGING NOT ENABLED — LOW

Enable user activity logging on all Redshift parameter groups.

L5 - ELASTIC LOAD BALANCER ACCESS LOGS NOT ENABLED — LOW

Enable access logging for all load balancers.

I1 - EC2 TERMINATION PROTECTION IS DISABLED — INFORMATIONAL

Consider enabling EC2 instance termination protection for all affected instances.

VULNERABILITY FINDINGS

The vulnerabilities below were identified and verified by Rhino Security Labs during the process of this AWS Post-Exploitation Assessment for Contoso. Retesting should be planned following the remediation of these vulnerabilities.

C1 Overly Permissive IAM Resource

Risk Rating: Critical

Exploitation Likelihood: **Critical** | Potential Impact: **Critical**

Description

The IAM permissions that were given to the affected IAM resource are overly permissive and allow for an escalation of privileges within the AWS environment.

When checking for this vulnerability, the assessor analyzes the permissions given to the affected IAM resource to determine if any permission or combination of permissions would allow that resource access to AWS APIs that are not intended to be accessible. Below, several AWS privilege escalation methods are listed, along with their potential impact.

Note: Not all methods below are applicable to the set of affected IAM resources, but are supplied for your information. The numbers next to the list of affected resources indicate which privilege escalation methods they are vulnerable to.

1. Creating a new policy version

Method: An attacker with the `iam:CreatePolicyVersion` permission can create a new version of a policy that they have access to. This allows them to define their own custom permissions. When creating a new policy version, it needs to be set as the default version to take effect, which would usually require the `iam:SetDefaultPolicyVersion` permission, but when creating a new policy version, it is possible to include a flag that will automatically create it as the new default version. That flag does **not** require the `iam:SetDefaultPolicyVersion` permission to use.

Potential Impact: With access to create an arbitrary policy version, a resource can specify a completely custom policy document, which means they can escalate to full administrator permissions.

2. Setting the default policy version to an existing version

Method: An attacker with the `iam:SetDefaultPolicyVersion` permission may be able to escalate privileges through existing policy versions that are not currently in use. 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.

Potential Impact: The potential impact is associated with the differences of permissions that the different policy versions allow.

3. Creating an EC2 instance with an existing instance profile

Method: An attacker with the `iam:PassRole` and `ec2:RunInstances` permissions can create a new EC2 instance that they will have operating system access to and pass an existing EC2 instance profile/service role to it. They can then login to the instance and request the associated AWS keys from the EC2 instance meta data, which gives them access to all of the permissions that the associated instance profile/service role has.

Potential Impact: This attack would grant access to any permission that existing EC2 service roles have access to.

4. Creating a new user access key

Method: An attacker with the `iam:CreateAccessKey` permission on other users is able to create an access key ID and secret access key belonging to another user in the AWS environment.

Potential Impact: This would give an attacker the same level of permissions as any user they were able to create an access key for.

5. Creating a new login profile

Method: An attacker with the `iam:CreateLoginProfile` permission on other users is able to create a password to use to login to the AWS console on any user that does not currently have a login profile setup.

Potential Impact: This would give an attacker the same level of permissions as any user they were able to create a login profile for.

6. Updating an existing login profile

Method: An attacker with the `iam:UpdateLoginProfile` permission on other users is able to change the password used to login to the AWS console on any user that already has a login profile setup.

Potential Impact: This would give an attacker the same level of permissions as any user they were able to update the login profile for.

7. Attaching a policy to a resource

Method: An attacker with the `iam:AttachUserPolicy`, `iam:AttachGroupPolicy`, or `iam:AttachRolePolicy` permissions can escalate privileges by attaching a policy to a user, group, or role that they have access to, adding the permissions of that policy to the attacker.

Potential Impact: It would be possible to attach the AdministratorAccess AWS managed policy, giving the attacker full administrator access to the environment.

8. Updating an inline policy for a resource

Method: An attacker with the `iam:PutUserPolicy`, `iam:PutGroupPolicy`, or `iam:PutRolePolicy` permissions can escalate privileges by updating an existing inline policy for a user, group, or role that they have access to, adding the permissions of that policy to the attacker.

Potential Impact: Due to the ability to specify an arbitrary policy document, the attacker could escalate to full administrator privileges this way.

9. Adding a user to a group

Method: An attacker with the `iam:AddUserToGroup` permission can use it to add themselves to an existing IAM Group in the AWS account.

Potential Impact: The attacker would be able to gain privileges of any existing group in the account.

10. Updating the AssumeRolePolicyDocument of a role

Method: An attacker with the `iam:UpdateAssumeRolePolicy` and `sts:AssumeRole` permissions would be able to change the assume role policy document of any existing role to allow them to assume that role.

Potential Impact: This would give the attacker the privileges that are attached to any role in the account.

11. Passing a role to a new Lambda function, then invoking it

Method: A user with the `iam:PassRole`, `lambda:CreateFunction`, and `lambda:InvokeFunction` permissions can escalate privileges by passing an existing IAM role to a new Lambda function that includes code to import the AWS library to their programming language of choice, then using it perform actions of their choice. It could then be run by invoking the function through the API.

Potential Impact: This would give a user access to the privileges associated with any Lambda service role that exists in the account.

12. Passing a role to a new Lambda function, then triggering it with DynamoDB

Method: A user with the `iam:PassRole`, `lambda:CreateFunction`, and `lambda:CreateEventSourceMapping` (and possibly `dynamodb:PutItem` and `dynamodb:CreateTable`) permissions, but without the `lambda:InvokeFunction` permission, can escalate privileges by passing an existing IAM role to a new Lambda function that includes code to import the AWS library to their programming language of choice, then using it perform actions of their choice. They then would need to either create a DynamoDB table or use an existing one, to create an event source mapping for the Lambda function pointing to that DynamoDB table. Then they would need to either put an item into the table or wait for another method to do so that the Lambda function will be invoked.

Potential Impact: This would give an attacker access to the privileges associated with any Lambda service role that exists in the account.

13. Updating the code of an existing Lambda function

Method: An attacker with the `lambda:UpdateFunctionCode` permission could update the code in an existing Lambda function with an IAM role attached so that it would import the AWS library in that programming language and use it to perform actions on behalf of the role. They would then need to wait for it to be invoked if they were not able to do so directly.

Potential Impact: This would give an attacker access to the privileges associated with the Lambda service role that is attached to that function.

14. Passing a role to a Glue Development Endpoint

Method: An attacker with the `iam:PassRole` and `glue:CreateDevEndpoint` permissions could create a new AWS Glue development endpoint and pass an existing service role to it. They then could SSH into the instance and use the AWS CLI to have access of the permissions the role has access to.

Potential Impact: This would give an attacker access to the privileges associated with any Glue service role that exists in the account.

15. Updating an existing Glue Dev Endpoint

Method: An attacker with the `glue:UpdateDevEndpoint` permission would be able to update the associated SSH public key of a Glue instance to then SSH into it and have access to the permissions the attached role has access to.

Potential Impact: This would give an attacker access to the privileges associated with the role attached to the specific Glue development endpoint.

16. Passing a role to CloudFormation

Method: An attacker with the `iam:PassRole` and `cloudformation:CreateStack` permissions would be able to escalate privileges by creating a CloudFormation template that will perform actions and create resources using the permissions of the role that was passed when creating a CloudFormation stack.

Potential Impact: This would give an attacker access to the privileges associated with the role that was passed when creating the CloudFormation stack.

17. Passing a role to Data Pipeline

Method: An attacker with the `iam:PassRole` and `datapipeline:CreatePipeline` permissions would be able to escalate privileges by creating a pipeline that will run an arbitrary AWS CLI command, either once or on an interval.

Potential Impact: This would give the attacker access to the privileges associated with the role that was passed when creating the pipeline.

Affected Services

IAM Groups (Name | Method Numbers)

cs-engineering | 5

contoso-bucket-monitor-r | 7, 8

cs-buckets-service | 1, 2, 4, 5, 6, 7, 8, 9

cs-data-infra-ml | 3

cs-paas-run | 3

cs-data-infra-elasticsearch | 3, 16

IAM Users (Name | Method Numbers)

data-infra-es-emr | 3, 16

jdoe | 3

vault-qa-aws-athena-mount | 1, 2, 4, 5, 6, 7, 8, 9

vault-prod-aws-athena-mount | 1, 2, 4, 5, 6, 7, 8, 9

contosmaker | 3, 4, 7, 8

paas-emr | 3, 16

mdoctor | 3, 16

axiomat | 5

cs-iam-legend | 1, 2, 4, 5, 6, 7, 8, 9

cs-external-account | 1, 2, 4, 5, 6, 7, 8, 9

Remediation

Ensure that the principle of least privilege is followed where possible. This means that any given user, role, or group should have only the permissions they require and use, but nothing more and that those permissions should be isolated to only resources that they need to interact with. Segregation of permissions can prevent or slow down an attacker from moving through an environment.

It is suggested to avoid using the wildcard character (*) in permissions or the resources they affect without more context. IAM resources should only be able to access permissions that they require and only be able to use those privileges on resources need to.

Sometimes it may not be possible to remove certain permissions from a given user, role, or group. In those cases, it should be noted what is possible with those permissions.

Testing Process

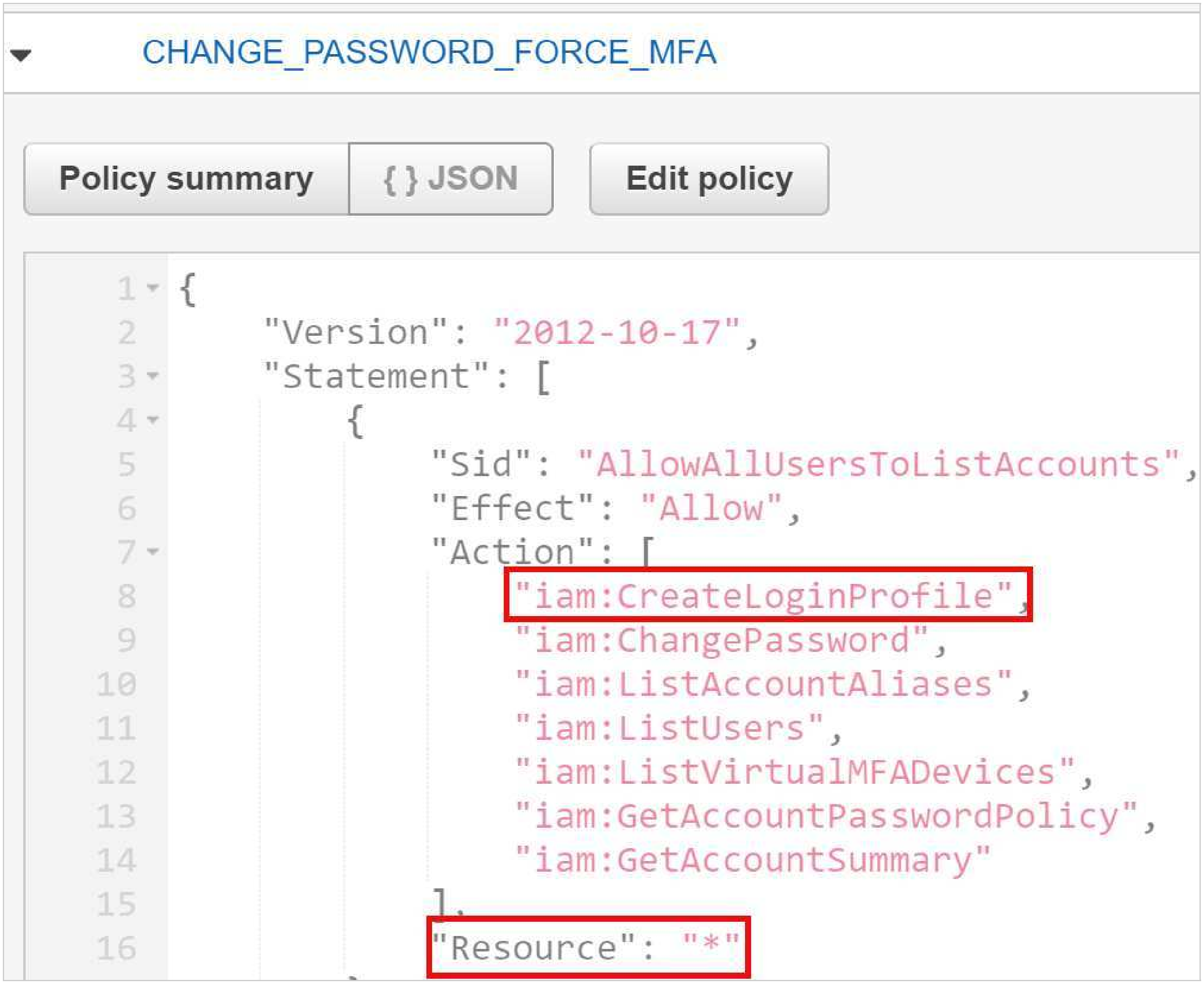
This vulnerability was discovered by using the privilege escalation module of an internally built AWS pentesting tool, Pacu. The module attempts to enumerate access of the current account and then attempts to escalate privileges through any available methods. Where full IAM read access is supplied or escalated to, the module can also be run against all users in the account to see which ones have the potential for privilege escalation.

Using the permissions supplied to the assessor at the beginning of the engagement, they were able to escalate to full administrator access. The user account had the `iam:CreateLoginProfile` permission with its allowed resources set to `"*"`. This allowed the assessor to create a login profile for any user in the account and login as them. Two users that had full IAM access and no passwords were found, those are listed here:

- arn:aws:iam::000000000000:user/vault-prod-aws-athena-mount
- arn:aws:iam::000000000000:user/vault-qa-aws-athena-mount

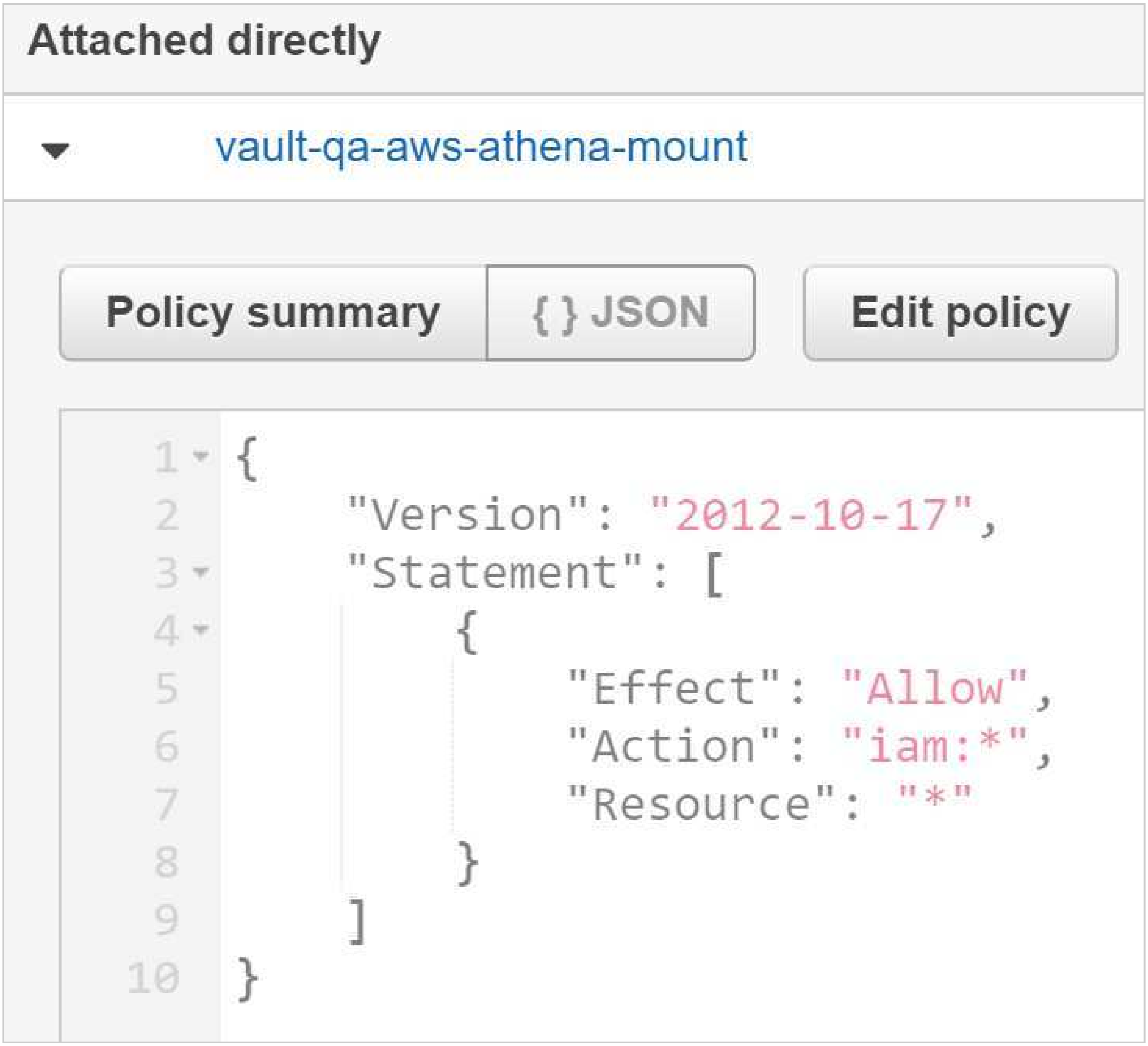
The assessor created a password for and logged in as the vault-ga-aws-athena-mount user, where they then attached an administrator policy to their original user.

This screenshot shows the policy `CHANGE_PASSWORD_FORCE_MFA` and the `iam:CreateLoginProfile` permission that was attached to the assessors user through a group membership.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllUsersToListAccounts",
            "Effect": "Allow",
            "Action": [
                "iam:CreateLoginProfile",
                "iam:ChangePassword",
                "iam:ListAccountAliases",
                "iam:ListUsers",
                "iam:ListVirtualMFADevices",
                "iam:GetAccountPasswordPolicy",
                "iam:GetAccountSummary"
            ],
            "Resource": "*"
```

The following screenshot shows the directly attached policy on the vault-qa-aws-athena-mount user, which was used to attach an administrator policy to the assessors account.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:*",
            "Resource": "*"
        }
    ]
}
```

Due to the fact that the assessor had full read access to the IAM service, all user accounts in the AWS account were checked for privilege escalation paths. The affected services section will list the applicable users and groups. Keep in mind that users listed under affected services only will include directly attached escalation methods or escalation methods from a combination of groups and policies and not any inherited from a group that was listed.

C2 Privilege Escalation to AWS Administrator

Risk Rating: Critical

Exploitation Likelihood: **Critical** | Potential Impact: **Critical**

Description

The assessor was able to escalate their access to the AWS environment through a chain of actions that led to information disclosures and privilege escalations that ultimately lead to full administrator access of the AWS account. That process is outlined below in the Testing Process section of this vulnerability.

Affected Services

S3 Buckets

contoso-devops

RDS Databases

rancher-mgmt | arn:aws:rds:us-east-1:000000000000:db:ranch

IAM Users

contoso | arn:aws:iam::000000000000:user/john.doe

contoso | arn:aws:iam::000000000000:user/contoso

EC2 Elastic Block Store Volumes

rancher-mgmt-ue1c | vol-5fce2f0b056348dc1 | us-east-1

EC2 Instances

staging-bastion - bastion1 | i-5efc2e2bcca120f1f | us-east-1

Remediation

Review the attack chain taken by the assessor and reevaluate the relevant access controls at each pivot point exploited by them.

In this specific case, we have the following recommendations:

- Don't store private SSH keys in S3 buckets that are accessible by all AWS users.
- Consider reevaluating the permissions that are given to developers in the AWS environment to prevent expanded

access through EBS volume copying or running EC2 instances in VPCs that shouldn't be accessed like they were.

- Where supported, always use the password prompt provided by Linux CLI tools rather than passing the password in as cleartext to prevent cleartext passwords from being written the .bash_history. An example relating to this scenario is when using the "mysql" command line tool, rather than passing in the password in using the "--password=" argument, just pass in the "-p" argument. After hitting enter, the "-p" argument will cause the program to prompt you with "Enter password: " which will protect the password from showing up on the screen and from being logged to .bash_history.
- Don't store AWS secret access keys in cleartext, even if they are stored in a database. As part of a log, just the access key ID is sufficient for detecting who performed the action that was logged and it does not disclose any private information

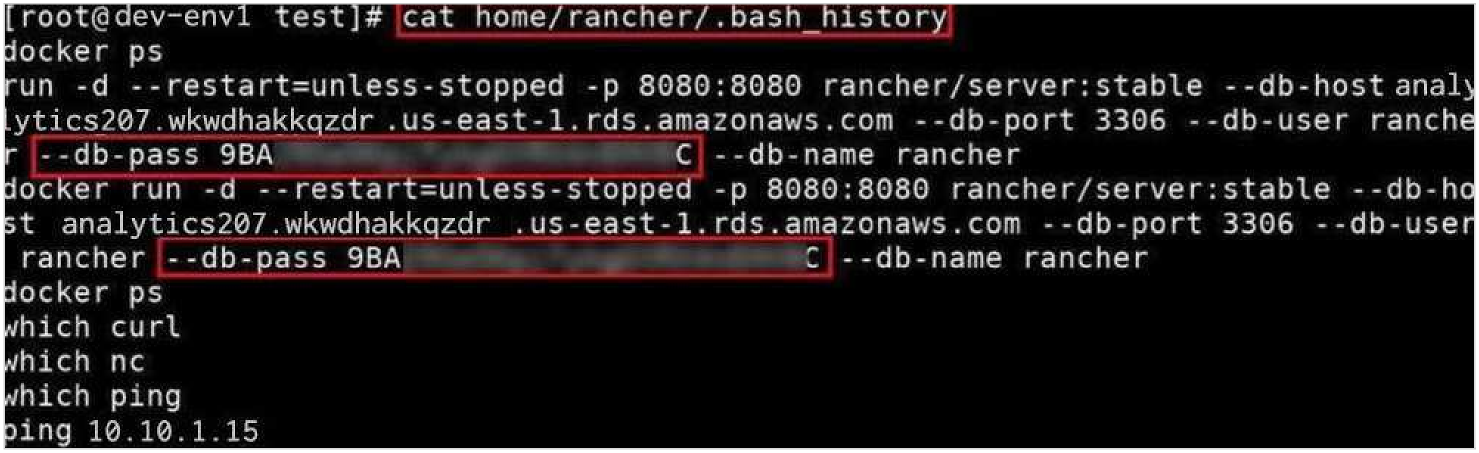
Testing Process
The secondary attack chain the assessor followed to gain administrator access can be reviewed below.

The process that the assessor followed to gain administrator access to the AWS account began with finding sensitive data in an S3 bucket in the account. Using the staging-bastion.pem SSH private key that was discovered, the assessor was able to access the EC2 instance "staging-bastion - bastion1" (i-0cba9e1bbbbb120f1f).

With access to that instance, the assessor then began creating snapshots of existing Elastic Block Store Volumes, then creating new volumes from those snapshots. In doing this, the assessor was able to copy and access EBS volumes without ever detaching them from their instances and disrupting service. With the newly created volumes, the assessor then attached them to the staging-bastion EC2 instance and mounted them to a folder they created so they could explore the data stored in each one(https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-using-volumes.html).

One EBS volume that the assessor copied and attached (vol-0f06ef0a8562989c9) to the staging-bastion instance included a small .bash_history file with sensitive information stored in cleartext. A command was found in the file that contained database credentials in cleartext for the "rancher-mgmt" RDS database.

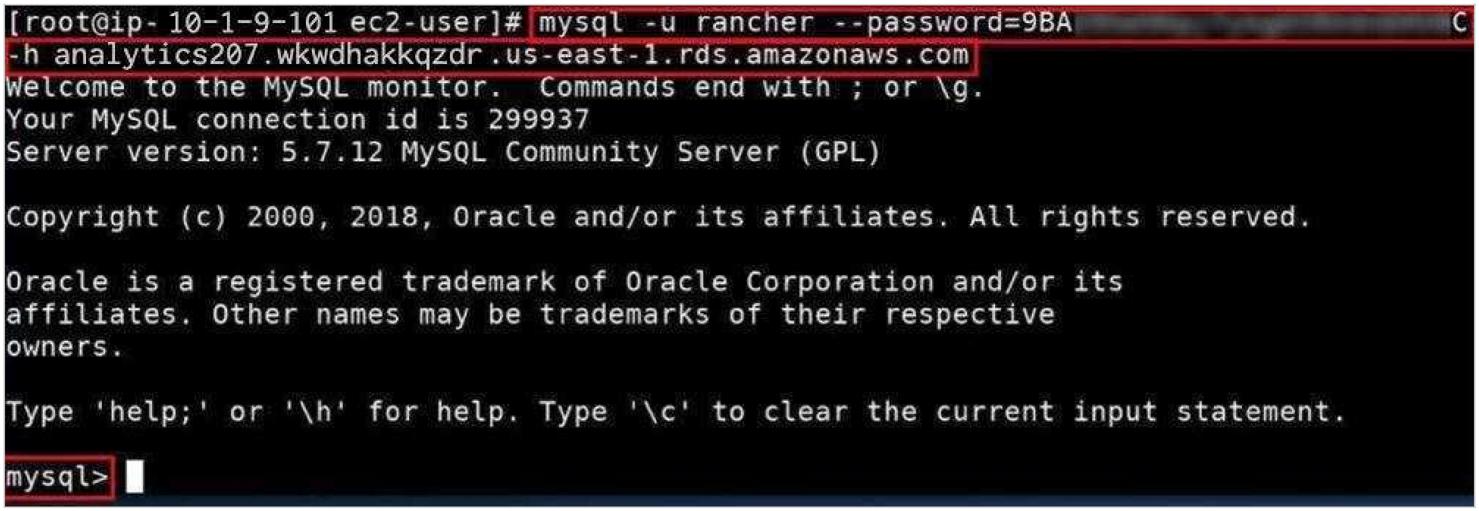
The contents of that .bash_history file can be seen in this screenshot:



```
[root@dev-env1 test]# cat home/rancher/.bash_history
docker ps
run -d --restart=unless-stopped -p 8080:8080 rancher/server:stable --db-host analy
lytics207.wkwdhakkqzdr .us-east-1.rds.amazonaws.com --db-port 3306 --db-user ranche
r --db-pass 9BA[redacted]C --db-name rancher
docker run -d --restart=unless-stopped -p 8080:8080 rancher/server:stable --db-ho
st analytics207.wkwdhakkqzdr .us-east-1.rds.amazonaws.com --db-port 3306 --db-user
 rancher --db-pass 9BA[redacted]C --db-name rancher
docker ps
which curl
which nc
which ping
ping 10.10.1.15
```

To reach this point, the assessor first created a snapshot of vol-0102ec1a8262989c9 (snap-0002c54fc6224943230), which was attached to the EC2 instance "rancher-mgmt-ue1c" (i-069f38ddb1e3d6b67). Then they created a new volume from that snapshot (vol-0c82fc5acd46454e2), to which they then attached it to the staging-bastion EC2 instance to access the files. The .bash_history file was located at /home/rancher/.bash_history.

Now at this point, the assessor had valid credentials to the rancher-mgmt RDS database, but no way to access the database to try and login due to the RDS and VPC settings. To work around this, the assessor launched an EC2 instance into a public subnet that was located in the same VPC as the rancher-mgmt RDS database. From this EC2 instance, the assessor was able to connect to the database successfully using the "mysql" command line tool. The screenshot below shows the assessor authenticating with the database.

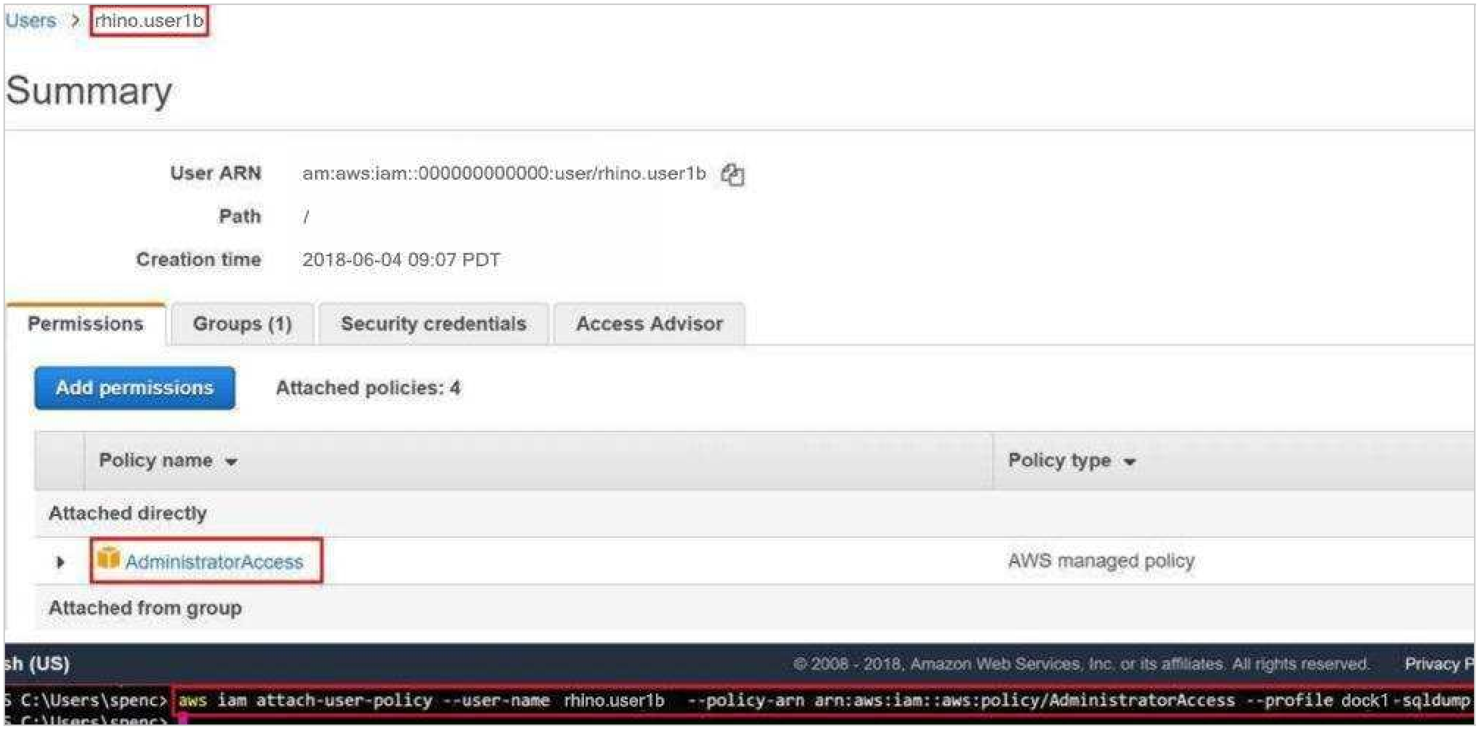


```
[root@ip-10-1-9-101 ec2-user]# mysql -u rancher --password=9BA[redacted]C
-h analytics207.wkwdhakkqzdr.us-east-1.rds.amazonaws.com
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 299937
Server version: 5.7.12 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Given this access, the assessor used the command line tool "mysqldump" to create a SQL dump of the entire "rancher"database. They then transferred this file back to their workstation to analyze the data. At this point, they terminated the EC2 instance that was run to access the RDS database.

While analyzing the data, the assessor discovered the MySQL table "audit_logs", which seemed to contain some very sensitive information, including an AWS access key ID and secret access key. By checking the access key ID in AWS IAM, it was discovered that it belonged to the IAM user john.doe(arn:aws:iam::0000000000:user/john.doe). john.doe had administrator access to the AWS environment, so the assessor abused those permissions to upgrade their own account to an AWS administrator.

The screenshot below shows the assessor using john.doe's AWS keys to attach the AWS managed Administrator Access policy to their own IAM user and the resulting attached policy in the AWS console.



Users > rhino.user1b

Summary

User ARN	arn:aws:iam::000000000000:user/rhino.user1b
Path	/
Creation time	2018-06-04 09:07 PDT

Attached policies: 4

Policy name	Policy type
Attached directly	
AdministratorAccess	AWS managed policy
Attached from group	

sh (US) © 2008 - 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy P

```
S C:\Users\spenc> aws iam attach-user-policy --user-name rhino.user1b  --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --profile dock1-sqldump
S C:\Users\spenc>
```

H1 EC2 User Data Sensitive Information Leakage

Risk Rating: High

Exploitation Likelihood: **High** | Potential Impact: **High**

Description
The User Data associated with different EC2 instances exposes private information. Any user with access to the EC2 console or any process with access to the instance metadata is able to retrieve these secrets.

Amazon recommends to not store sensitive data like passwords or secrets in EC2 User Data. For more information, visit the following link on Metadata and User Data:

- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html

Affected Services

EC2 Instances

twilight-fog | i-f367e543 | us-east-1

dawn-bird | i-234e5fd4 | us-east-1

gentle-water | i-64a9ca3e2 | us-east-1

gentle-bird | i-a43df819 | us-east-1

nameless-mud | i-51cf2fe1 | us-east-1

Remediation
Remove hardcoded secrets and passwords from EC2 User Data. As EC2 User Data is static, unencrypted, and viewable by users with the correct EC2 permissions, it is suggested to not hardcode sensitive values into the data. Amazon recommends to not store any passwords or secrets in EC2 User Data as well.

One method of encrypting EC2 User Data can be found in this short blog post:

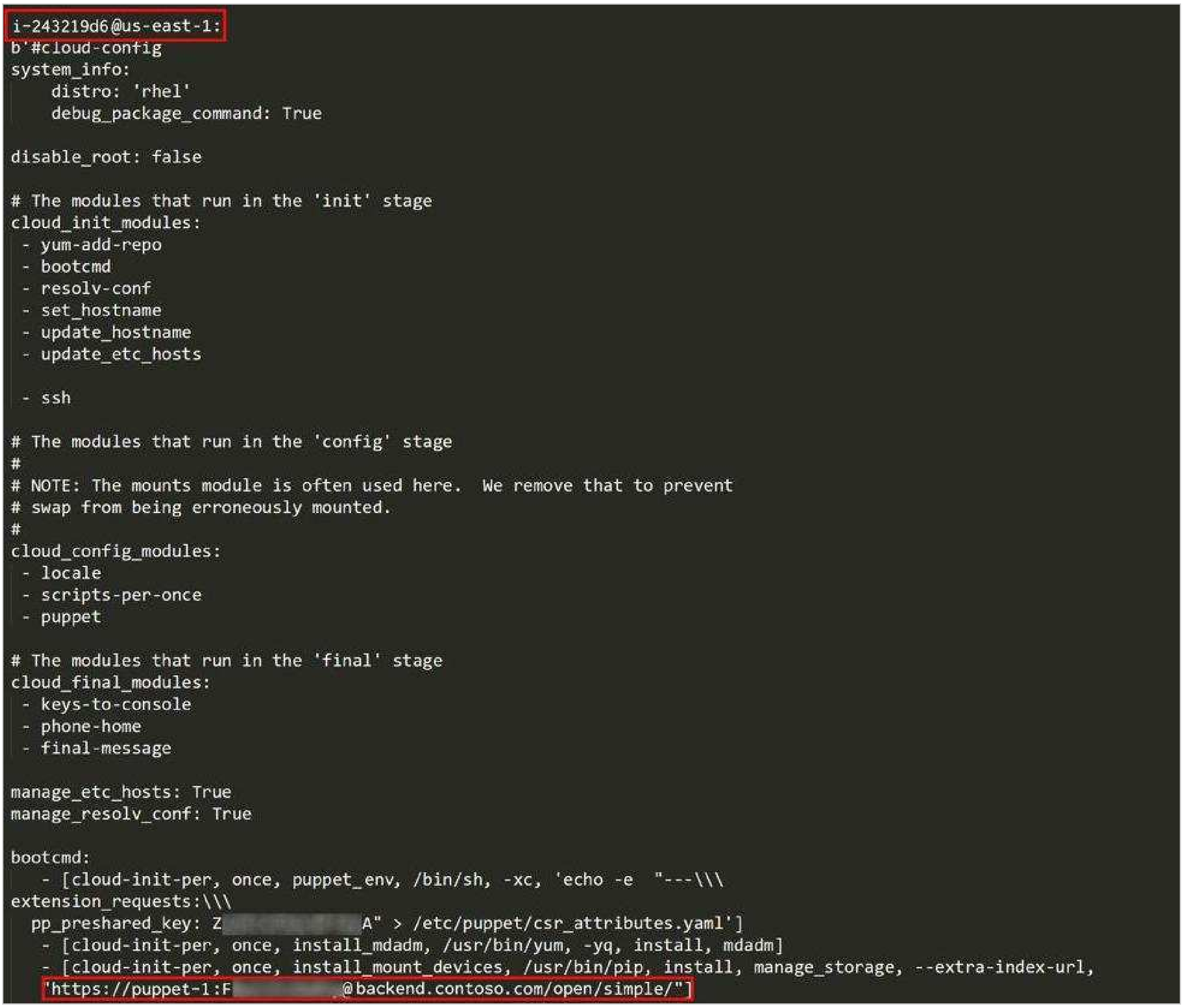
- https://www.whaletech.co/2015/02/25/securing-ec2-user-data.html

Testing Process
This vulnerability was discovered by using a script that the assessor wrote to scrape and decode all user data associated

with any EC2 instances in the account, then by manually checking for sensitive information hardcoded in.

The data that was discovered was a username and password for `https://private.contoso.com/pypi/simple/` and was found to be the same across each of the affected instances.

The following screenshot shows the user data extracted from one of the affected instances.



```
i-243219d6@us-east-1:
b'#cloud-config
system_info:
    distro: 'rhel'
    debug_package_command: True

disable_root: false

# The modules that run in the 'init' stage
cloud_init_modules:
 - yum-add-repo
 - bootcmd
 - resolv-conf
 - set_hostname
 - update_hostname
 - update_etc_hosts

 - ssh

# The modules that run in the 'config' stage
#
# NOTE: The mounts module is often used here.  We remove that to prevent
# swap from being erroneously mounted.
#
cloud_config_modules:
 - locale
 - scripts-per-once
 - puppet

# The modules that run in the 'final' stage
cloud_final_modules:
 - keys-to-console
 - phone-home
 - final-message

manage_etc_hosts: True
manage_resolv_conf: True

bootcmd:
   - [cloud-init-per, once, puppet_env, /bin/sh, -xc, 'echo -e  "---\\\
extension_requests:\\\
  pp_preshared_key: Z[redacted]A" > /etc/puppet/csr_attributes.yaml']
   - [cloud-init-per, once, install_mdadm, /usr/bin/yum, -yq, install, mdadm]
   - [cloud-init-per, once, install_mount_devices, /usr/bin/pip, install, manage_storage, --extra-index-url,
   'https://puppet-1:F[redacted]@backend.contoso.com/open/simple/"]
```

H2 CloudTrail Logging Disabled

Risk Rating: High

Exploitation Likelihood: **Medium** | Potential Impact: **Medium**

Description

There are no active CloudTrail instances in place. CloudTrail is an important part of keeping track of the history of different API calls being made within the AWS account. By default, the CloudTrail service will keep a log history of the past 90 days for a region; this can be seen by visiting the "Event history" tab of the CloudTrail console, but this kind of logging does not permit long term storage, encryption, or some API calls and it is region-specific, rather than across all regions.

For more information, visit the following link:

- https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-getting-started.html

Affected Services

Affected Services

Cloudtrail

Remediation

The default CloudTrail "Event history" holds the past 90 days of activity, but that activity is region specific, temporary, unencrypted, and is missing some API calls that an actual trail would catch. It is suggested that a CloudTrail instance be created and applied to all regions, to log all actions across the entire AWS account. Log file validation and encryption should be utilized to ensure at-rest safety and integrity of the information held in them.

Where applicable, it is also recommended to track S3 object-level API activity and Lambda function invocation activity.

For more information on the difference between API calls that are covered by the CloudTrail Event History and CloudTrail instances please see the following pages.

CloudTrail Event History:

- https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events-supported-services.html

CloudTrail:

- https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-supported-services.html

Testing Process

This vulnerability was discovered by noting no active trails were in use through the CloudTrail API.

H3 AWS S3 Bucket Data Leakage

Risk Rating: High

Exploitation Likelihood: **Medium** | Potential Impact: **High**

Description

AWS S3 bucket data leakage occurs when there is a misconfiguration of permissions for that bucket. Some common misconfigurations include public read access for files and public listing access for the buckets themselves. This can allow people to list the files in the bucket, as well as read the contents of them. An attacker could potentially exploit “write” access to a bucket to add a key logger to a file that the application includes on its pages, such as a jQuery file. Most of the time, "write" access is accompanied by "delete" access, which could allow an attacker to wipe out the contents of the bucket.

Affected Services

S3 Buckets

contoso-devops

Remediation

Restrict bucket permissions to be accessible by users or servers who need access to its resources. By default, AWS S3 buckets permissions are set up to be very restrictive. It is important to remember that “Everyone” in IAM permissions literally means everyone on the Internet, and would immediately expose the full set of data to anyone who requests it. Confirm that only people/groups that specifically need to are able to “list” or “write” the storage buckets. It is best practice to only allow the bare minimum permissions for different users who need access. It should also be confirmed that for individual files in bucket, only people/groups that are specified are able to access those files.

Testing Process

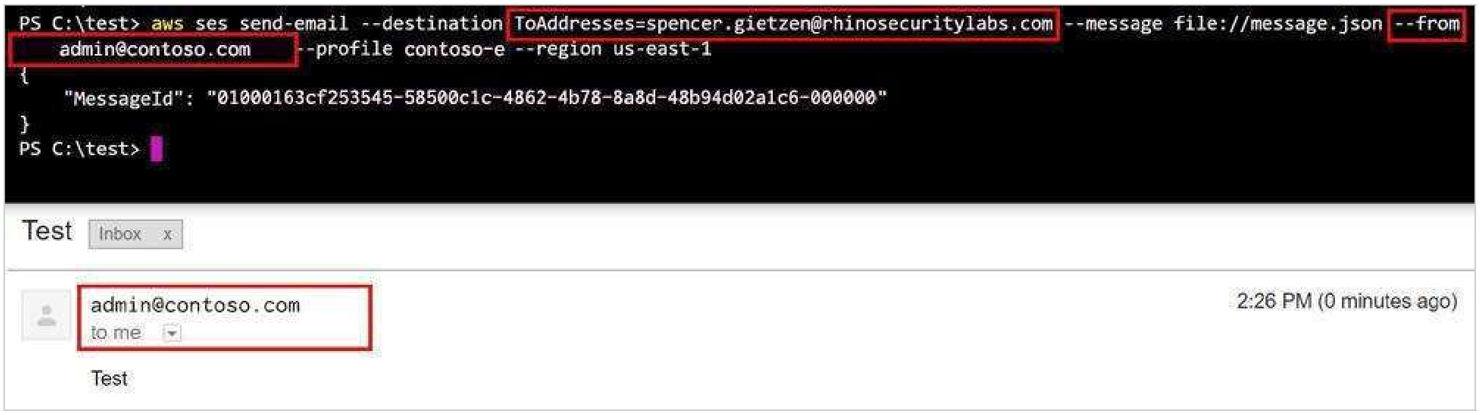
This vulnerability was discovered by reviewing the contents of each S3 bucket that the assessor had access to.

The assessor discovered many credentials relating to the web servers and associated services and applications. These sensitive files were located at s3://contoso-devops/rancher/*, where there were multiple files for each environment setup for the web applications. The assessor was able to find AWS keys, database passwords, and web app credentials contained within these.

Using the AWS access key ID and secret access key that was discovered, we checked the permissions and found that they had full access to AWS SES. Using those keys, the assessor was able to send emails out originating from any of the following emails:

- admin@contoso.com
- no-reply@contoso.com
- support@contoso.com

The screenshot below shows the assessor sending an email to himself from admin@contoso.com



```
PS C:\test> aws ses send-email --destination ToAddresses=spencer.gietzen@rhinosecuritylabs.com --message file://message.json --from
    admin@contoso.com    --profile contoso-e --region us-east-1
{
    "MessageId": "01000163cf253545-58500c1c-4862-4b78-8a8d-48b94d02a1c6-000000"
}
PS C:\test>
```

Test Inbox x

admin@contoso.com
to me
2:26 PM (0 minutes ago)

Test

H4 Weak IAM Password Policy

Risk Rating: High

Exploitation Likelihood: **High** | Potential Impact: **High**

Description

The IAM password policy is not sufficient to keep the authentication process secure. Weak or non-existent password policies allow users to create sub-par passwords that can easily be cracked from a hash, brute-forced, or even simply guessed.

See Also:

- https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_passwords_account-policy.html?icmpid=docs_iam_console

Affected Services

AWS Account ID

000000000000

Remediation

Implement a password policy that ensures strong, high-entropy passwords for all IAM users. It should be required that passwords contain a minimum of 12 characters, including at least one lowercase letter, one uppercase letter, one number, and one special character. It is also suggested that passwords expire somewhere between 30 to 90 days. Password reuse should be disabled to prevent old, compromised passwords from impacting the security of the account.

Testing Process

This vulnerability was discovered by visiting the IAM Password Policy page and noting password reuse was allowed and that passwords do not expire.

M1 Redshift Cluster Database Encryption Disabled

Risk Rating: Medium

Exploitation Likelihood: **Medium** | Potential Impact: **Medium**

Description

Redshift Cluster Databases were discovered that do not have encryption enabled. Enabling encryption will keep the database data and metadata encrypted at rest for itself and all snapshots taken from it.

See Also:

- https://docs.aws.amazon.com/redshift/latest/mgmt/working-with-db-encryption.html

Affected Services

Redshift Clusters (Name | Region)

email-anti-abuse | us-east-1

sales-email | us-east-1

Redshift Clusters (Name | Region)

email-anti-abuse | us-east-1

sales-email | us-east-1

Remediation

Enable encryption for all affected databases. The encryption setting for databases is an immutable setting, so to enable encryption for existing databases, it is required to unload the data from the existing database and load it into a newly created database with encryption enabled.

For more information on migrating from an unencrypted cluster to an encrypted cluster, visit the following link:

- https://docs.aws.amazon.com/redshift/latest/mgmt/migrating-to-an-encrypted-cluster.html

Testing Process

This was confirmed by inspecting the affected clusters and noting that the encryption setting was not enabled.

M2 VPC Flow Logs Disabled

Risk Rating: Medium

Exploitation Likelihood: **Medium** | Potential Impact: **Medium**

Description
VPC flow logs capture information about the IP traffic going to and from network interfaces in a VPC. Flow log data is stored using Amazon CloudWatch Logs. After you have created a flow log, you can view and retrieve its data in Amazon CloudWatch Logs.

Flow logs can have many uses, such as diagnosing connection issues to an instance in the VPC or allowing easy inspection of traffic in response to a security incident.

See Also:
- https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/flow-logs.html

Affected Services
VPC Subnets (ID | VPC ID | Region)

subnet-01d23668 | vpc-97648ef2 | ap-northeast-2

subnet-eed7f6a4 | vpc-9a782fe | ap-northeast-2

subnet-1ee80977 | vpc-092c9165 | ap-south-1

subnet-40ad8e0a | vpc-0f618255 | ap-south-1

subnet-1838c971 | vpc-7ec5417 | ca-central-1

Remediation
Enable flow logs on the affected EC2 network interfaces.

Testing Process
This vulnerability was discovered by noting the lack of flow logs associated with the affected EC2 network interfaces.

M3 Redshift Parameter Group SSL Not Required

Risk Rating: Medium

Exploitation Likelihood: **Medium** | Potential Impact: **Medium**

Description

Amazon Redshift supports Secure Sockets Layer (SSL) connections to encrypt data and server certificates to validate the server certificate that the client connects to. This setting can be enabled to enforce secure connections and block plaintext connections attempting to be made.

For more information, visit the following link:

- https://docs.aws.amazon.com/redshift/latest/mgmt/connecting-ssl-support.html

Affected Services

Redshift Parameter Groups (Name | Region)

contosoanalytics | us-east-1

dimensions-71| us-east-1

dimensions-17 | us-east-1

default.redshift-1.0 | us-east-1

Remediation

Require SSL for all connections. This can be done by altering the setting `require_ssl` to true for each affected parameter group in AWS Redshift.

For more information, visit the following link:

- https://docs.aws.amazon.com/redshift/latest/mgmt/connecting-ssl-support.html

Testing Process

This vulnerability was discovered by analyzing the settings for the affected Redshift parameter groups.

M4 No IAM User Access Key Rotation

Risk Rating: Medium

Exploitation Likelihood: **Medium** | Potential Impact: **Medium**

Description

There is no plan in place to rotate IAM user access keys at a regular interval. By regularly rotating access keys, the chances of a set of keys being breached goes down significantly. Overtime, keys usually end up being used in more and more different places; this could be due to new computers, new servers, new employees, new applications, or other similar things. This can increase the chance that someone malicious may discover them and abuse them.

Affected Services

IAM Users

Remediation

Implement a plan to regularly rotate all IAM user access keys. Access keys should be treated the same as passwords in the sense that they need to expire for each user. It is suggested to rotate keys at least every 90 days, but 30 to 60 days would be even more secure.

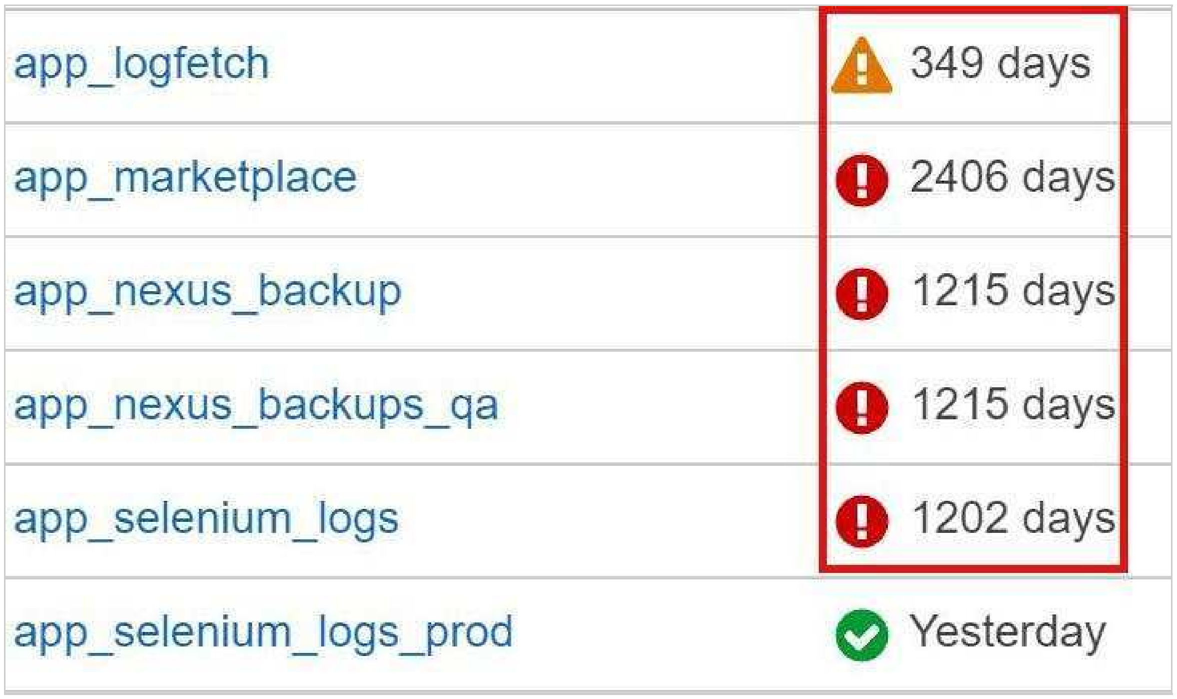
For an explanation on how to go about rotating access keys, visit this short blog post on the AWS security blog:

- https://aws.amazon.com/blogs/security/how-to-rotate-access-keys-for-iam-users/

Testing Process

This vulnerability was discovered by reviewing the "Access Key Age" setting for each IAM user. This indicated a large amount of time since the last key rotation was done.

The screenshot on the following page shows a subset of users with long-lasting access keys; some over 1000 days old and one over 2400 days old.



app_logfetch	⚠ 349 days
app_marketplace	❗ 2406 days
app_nexus_backup	❗ 1215 days
app_nexus_backups_qa	❗ 1215 days
app_selenium_logs	❗ 1202 days
app_selenium_logs_prod	✓ Yesterday

L1 Unencrypted Elastic Block Store (EBS) Snapshots

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Medium**

Description

The account has Elastic Block Store (EBS) snapshots without encryption enabled. Encryption allows that data to be stored securely and adds another layer of defense against attackers.

By enabling encryption, you are protecting:

- Data at rest
- All volumes created from those snapshots

Affected Services

EBS Snapshots (Snap ID | Region)

snap-12320a | us-east-1

snap-12301a | us-east-1

snap-122312a | us-east-1

snap-123b0a | us-east-1

snap-12300a | us-east-1

Remediation

Enable encryption for all EBS volumes, so that all future EBS snapshots will also be encrypted. EBS snapshot encryption is automatically done when creating a snapshot of an encrypted EBS volume, so it is best practice to encrypt all volumes, thus encrypting all snapshots.

Testing Process

This vulnerability was discovered by reviewing the list of EBS snapshots and noting the ones that were not encrypted.

L2 S3 Bucket Access Logging Not Enabled

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Medium**

Description

Access logging is not enabled for the affected S3 buckets. Access logging enables the ability to track requests for access to the bucket. The logs provide details about the incoming request and how it was responded to. There are no additional fees for enabled access logging, but the normal S3 storage charges occur for the stored logs.

For more information, visit the following link:

- https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html

Affected Services

S3 Buckets

contoso-artifacts

contoso-webpages

contoso-ml-data

contoso-messages

contoso-mothership-tasks

contoso-graph-prod

contoso-live-logs-qa

contoso-product

Remediation

Enable access logging for the affected S3 buckets. This can be done through the web console by going to the options of an affected bucket, then the `Properties` tab, then clicking the `Server access logging` button.

Testing Process

This vulnerability was discovered by reviewing the settings associated with the affected buckets and noting that access logging was disabled.

L3 S3 Bucket Versioning Not Enabled

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Low**

Description

Versioning enables a bucket to keep multiple versions of an object in it. Versioning is beneficial because it protects objects from unintended overwrites and deletions, as well as allows a simple way to archive objects. This is useful from a security perspective in the case that an attacker was able to alter or delete files, it is possible to recover the previous version and differentiate between the new and old versions.

For more information, visit the following link:

- https://docs.aws.amazon.com/AmazonS3/latest/dev/ObjectVersioning.html

Affected Services

S3 Buckets

contoso-artifacts

contoso-webpages

contoso-ml-data

contoso-messages

contoso-mothership-tasks

contoso-graph-prod

contoso-live-logs-qa

contoso-product

Remediation

Enable versioning for the affected buckets. This can be done through the web console by going to the options of an affected bucket, then the `Properties` tab, then clicking the `Versioning` button.

For more information, visit the following link:

- https://docs.aws.amazon.com/AmazonS3/latest/user-guide/enable-versioning.html

Testing Process

The vulnerability was discovered by reviewing the settings of the affected S3 buckets and noting that versioning was disabled.

L4 Redshift User Activity Logging Not Enabled

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Medium**

Description

Redshift user activity logging is not enabled. These logs are stored in AWS S3 and help manage security monitoring of the databases.

Redshift includes three kinds of information in each audit log:

- Connection logs (authentication, connections, disconnections)
- User logs (changes to database user definitions)
- User activity logs (each query run on the database)

If audit logging is enabled for a database instance, connection and user logs are enabled, but for user activity logging to be enabled, it must be setup specifically for each instance.

See Also:

- https://docs.aws.amazon.com/redshift/latest/mgmt/db-auditing.html

Affected Services

Redshift Parameter Groups (Name | Region)

contosoanalytics | us-east-1

dimensions-71| us-east-1

dimensions-17 | us-east-1

default.redshift-1.0 | us-east-1

Remediation

Enable user activity logging on all Redshift parameter groups. This can be done by editing the settings for the affected Redshift parameter group and enabling the `enable_user_activity_logging` setting, which will enable user activity logging for the group.

For more information, visit the following link:

- https://docs.aws.amazon.com/redshift/latest/mgmt/db-auditing.html#db-auditing-enable-logging

Testing Process

This vulnerability was discovered by analyzing the settings associated with the Redshift parameter group that was in use and noting the `enable_user_activity_logging` setting being disabled.

L5 Elastic Load Balancer Access Logs Not Enabled

Risk Rating: Low

Exploitation Likelihood: **Low** | Potential Impact: **Medium**

Description

Elastic Load Balancer access logs provide details on individual requests made. These logs include information on the request from the client and the response from the server, which allows analysis of traffic patterns and issues.

Access logs are stored in an S3 bucket of the users choice. There is no charge for enabling access logs, besides the actual S3 costs accumulated from the amount of stored data in the bucket.

For more information, visit the following link:

- https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/access-log-collection.html?icmpid=docs_elb_console

Affected Services

Elastic Load Balancers (Name | VPC | Region)

mesos-contosobteamqa-com | ibf13 | us-east-1

mesos-cosprocjava-ct-sites-qa | ife23 | us-east-1

mesos-onboarding-contosoqa-com | ied51 | us-east-1

build-public-contosoqa-com | ifc02 | us-east-1

build-contosoqa-com | iad53 | us-east-1

Remediation

Enable access logging for all load balancers. This can be done by creating an S3 bucket to store the logs, adding the associated policy statement to manage permissions between the load balancer and S3, then enabling access logs on each load balancer that is affected.

For more information, visit the following link:

- https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/enable-access-logs.html

Testing Process

This vulnerability was discovered by checking the access log attribute associated with the affected load balancers and noting that it was disabled.

I1 EC2 Termination Protection Is Disabled

Risk Rating: **Informational**

Exploitation Likelihood: **Informational** | Potential Impact: **Informational**

Description

EC2 instance termination protection prevents instances from being terminated from the AWS console, API, or CLI. This is used to prevent accidental or inappropriate instance termination by a user that doesn't have local access to the operating system. With termination protection enabled, instances can still be terminated locally, by a user with operating system access, or termination protection can be disabled and then the instance can be terminated through regular means.

For more information, visit the following link:

- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/terminating-instances.html#Using_ChangingDisableAPITermination

Affected Services

(Name | Instance ID | Region)

north-giraffe | i-0adbfff21sq3a7f9 | us-east-1

(Name | Instance ID | Region)

south-giraffe | i-0adbee1221sq3a7f9 | us-east-1

(Name | Instance ID | Region)

rough-fish | i-0adbef3221sq3a7f9 | us-east-1

Remediation

Consider enabling EC2 instance termination protection for all affected instances. This can be done through the EC2 console settings for each individual instance and in the launch configuration of new instances.

Testing Process

This vulnerability was discovered by reviewing the termination protection setting for the affected EC2 instances.

RHINO SECURITY LABS TOOLKIT

The software and tools used for security analysis are constantly evolving and changing. To stay at the forefront of industry trends, Rhino Security Labs regularly updates and integrates new tools into its AWS Post-Exploitation Assessment methodology. Below is the toolset our consultants use during a Web Application assessment.

Pacu

Pacu is a proprietary tool written by Rhino Security Labs for post exploitation of AWS environments. It assists the assessor in using compromised access keys to identify, abuse, and log AWS functions and misconfigurations. Piranha is a compilation of many research techniques combined into a single tool that creates a more effective and efficient framework for testing against AWS environments in a post exploitation scenario.

Scout2

Scout2 is an open source tool that helps assess the security posture of AWS environments. Using the AWS API, the Scout2 Python scripts fetch CloudTrail, EC2, IAM, RDS, and S3 configuration data. Helpful in early reconnaissance of AWS environments.

Prowler

Prowler is an open source tool for auditing and hardening configurations of AWS environments. It primarily focuses on checking for compliance in regards to the CIS Amazon Web Services Foundations Benchmark 1.1, but includes other custom checks as well.

CloudGoat

CloudGoat is a proprietary tool written by Rhino Security Labs that launches misconfigured AWS resources into an account, for the purpose of testing vulnerabilities and learning the process. This tool is used as a sandbox environment to test various attacks on, prior to performing them on a production AWS environment. This ensures safety for all actions taken within the AWS environment we were tasked with assessing.

PacuProxy

PacuProxy is a proprietary tool written by Rhino Security Labs and is directly integrated to the internally-built tool, Pacu. PacuProxy is a command and control (C2) framework specifically for attacking AWS environments.

Custom Scripts and Applications

In addition to the above tools, Rhino Security Labs also makes use of its own proprietary tools and scripts to quickly adapt to new and unique environments.

APPENDIX A: CHANGES TO ENVIRONMENT

The following changes were made to the environment in scope. These do not necessarily represent a significant impact to the environment, but are included for the full accounting of modifications by the penetration testing team at Rhino Security Labs.

NO CHANGES

No changes were made to the environment in scope, such as creating new user accounts or running new instances. This is provided as the full accounting of modifications by the penetration testing team at Rhino Security Labs.

888.944.8679
info@rhinosecuritylabs.com
464 12th Ave, Suite 300 | Seattle, WA 98122