SecurityInnovation®
THE SOFTWARE SECURITY COMPANY

MetaCorp
AWS Configuration Review
Final Report

Document Name:	Final Report
Date:	Sept. 1, 2020
Customer Contact:	John Smith <john.smith@example.com>
Author:	Brandon Cooper <bcooper@securityinnovation.com>
Project Manager:	Garrett Jaynes <gjaynes@securityinnovation.com>

SecurityInnovation®
THE SOFTWARE SECURITY COMPANY
BOSTON | SEATTLE | PUNE
187 Ballardvale St., Suite A195 ● Wilmington, MA 01887 ● Ph: +1.978.694.1008
getsecure@securityinnovation.com ● www.securityinnovation.com

Contact Information

Security Innovation

Business Contact

Jeff Berry
VP of Global Sales
jberry@securityinnovation.com
Mobile: +1.425.273.5607

Technical Contact

Joe Basirico
Senior VP of Engineering
jbasirico@securityinnovation.com
Mobile: +1.206.227.6458

Project Management Contact

Garrett Jaynes
Senior Project Manager
gjaynes@securityinnovation.com

MetaCorp

John Smith
VP of Marketing
john.smith@example.com

Executive Summary

Security Innovation performed an Amazon Web Services (AWS) Configuration review on behalf of MetaCorp between April 22 and May 2 (10 engineer days). This report summarizes the issues that were uncovered.

MetaCorp relies on AWS for infrastructure that supports MetaCorp's products, including their COWS (Corporate Onboarding Web Service) platform. MetaCorp has one master account for their organization, and two other AWS accounts linked to the master account for specific purposes. The AWS accounts within scope of this review were:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

The review is mainly focused on the security of the services used by MetaCorp, the security of MetaCorp's Virtual Private Cloud, and addressing any concerns about a breach in the perimeter of MetaCorp's infrastructure.

A wide range of security issues were discovered during the course of this review. Overall, it was evident that efforts were made in securing the many services used by MetaCorp. The issues that Security Innovation have discovered range in impact from potentially acceptable risks, in the case of Disabled RDS Backups, to critical vulnerabilities, in the case of IAM Policies with Permissive AssumeRole. The latter issue allows users outside of the organization to escalate privileges to that of an administrator for MetaCorp's AWS accounts, making this a vulnerability that requires immediate attention.

Sample report note - During this Cloud Configuration Review engagement 19 problems were identified, this has been reduced to 5 in this sample report for the sake of brevity.

Major observations are as follows:

- A total of 5 security issues were identified:
  - PR 1 - Unencrypted Data at Rest
  - PR 2 - AWS Account Misconfiguration - Logging Disabled for Multiple Services
  - PR 3 - AWS Users Without Multi-Factor Authentication
  - PR 4 - AWS Insecure Network ACLs
  - PR 5 - IAM Policies with Permissive AssumeRole
- The most common vulnerability is Logging Disabled for Multiple Services. The most severe vulnerability is IAM Policies with Permissive AssumeRole.
- From a STRIDE perspective, issues were found from the **S**poofing, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, and **E**levation of Privilege categories.
- The consequence of not immediately addressing the Critical-severity vulnerability (PR 5 IAM Policies with Permissive AssumeRole) is that the AWS accounts will be exposed to the risk of a full compromise - an anonymous attacker would be able to perform any operation while assuming the role of an admin.

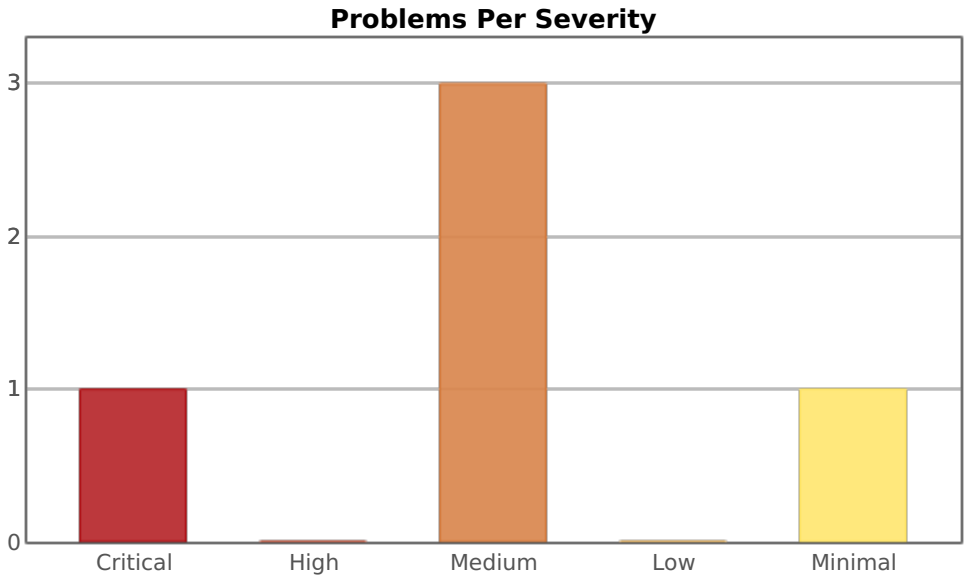
Introduction

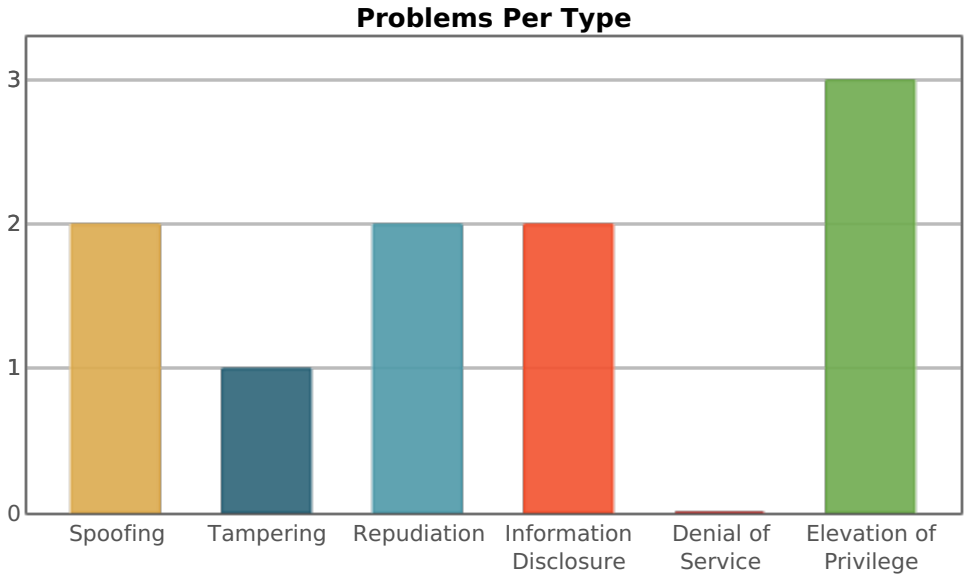
This report provides a summary of the findings discovered during the assessment. Each section is briefly described below:

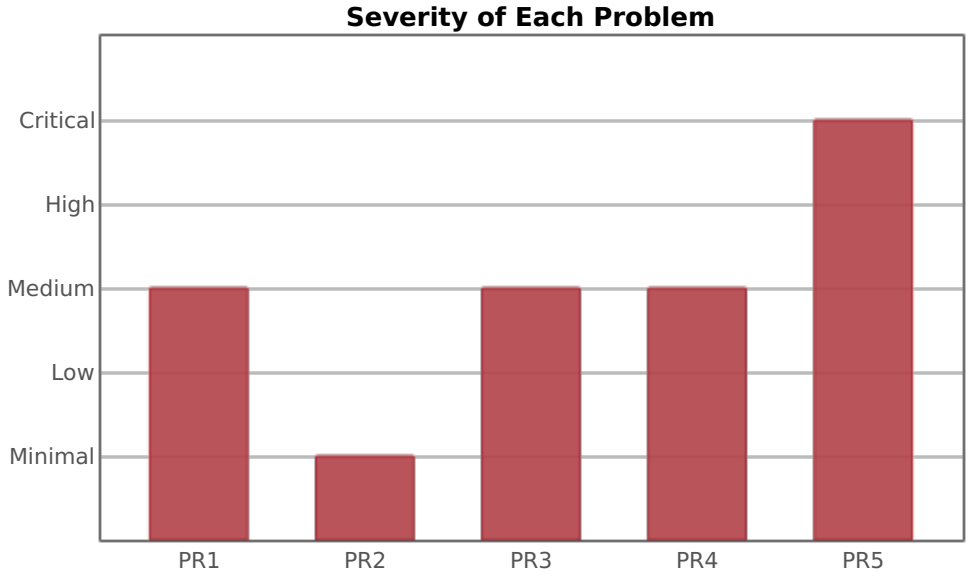
- The **Problem Report Summary** section summarizes the issues discovered during the engagement.
- The **Problem Reports** section contains the full text of each verified finding.
- The **Observations** section describes ad hoc observations and conclusions about the security of the controls based on the testing performed.
- The **Executed Test Cases** section lists a variety of performed tests and their observed results.
- The **Tools** section details the tools used during testing.
- The **Recommended Next Steps** section provides our recommendations for additional future testing of this system.

Problem Report Summary

A total of 5 problems were identified. This section describes, at a high level, each of the problems discovered. See the Problem Summaries section for a table of each problem discovered, its severity, description and consequences. The following charts display the number of problems for each level of severity, the number of problems for each STRIDE type (note that a problem may have more than one type), and each problem's overall severity.







Problem Summaries

The problem report summaries are sorted by problem report ID. The format of the problem report table is as follows:

- The problem report ID
- The component in which the issue was discovered
- The severity of the issue
- A short description of the issue
- The consequences of the issue

PR #	Component	Severity	Description	Consequence
1	AWS Account	Medium	Some of the Elastic Block Storage (EBS) volumes, RDS instances, and Redshift clusters used in the AWS accounts are not encrypted at rest using AWS Key Management Service (KMS).	An attacker with access to the internal AWS network or physical system may be able to access the data stored on these services.
2	AWS Account	Minimal	Logging is disabled for ELB, Redshift, S3, and VPC subnets.	Malicious activities performed by an attacker may remain untracked or undiscovered.

PR #	Component	Severity	Description	Consequence
3	AWS Account	Medium	Some users in the AWS account with password access do not have multi-factor authentication enabled.	An attacker that can compromise or brute force an AWS account user's password can log into the AWS account and carry out actions as that user.
4	AWS Account	Medium	Network ACLs in the AWS account configured with insecure traffic rules.	Network ACLs serve as an important defense in depth security control in the event insecure Security Groups are used. Insecure Network ACLs increase the attack surface of EC2 instances within a Subnet by potentially exposing ports and services to external attackers.
5	AWS Account	Critical	IAM Policies in the AWS account have permissive `sts:AssumeRole` rules.	IAM Users, Groups, and Roles that have these IAM Policies attached to them may be able to elevate their privileges to access unintended AWS functionality.

Problem Reports

Below are the complete Problem Reports for all discovered issues.

Problem Report 1 - Unencrypted Data at Rest

Some of the Elastic Block Storage (EBS) volumes, Relational Database Service (RDS) instances, and Redshift clusters used in the AWS accounts are not encrypted at rest using AWS Key Management Service (KMS). An attacker with access to the internal AWS network or physical system may be able to access the data stored on these services.

Component	AWS Account
STRIDE	Information Disclosure
CWE	CWE-311: Missing Encryption of Sensitive Data
CVSS v2 Score	2.3 (AV:A/AC:M/Au:S/C:P/I:N/A:N)
CVSS v3 Score	2.4 CVSS:3.0/AV:A/AC:L/PR:H/UI:N/S:U/C:L/I:N/A:N

Overall Severity	**Medium**
Vulnerability Type	**Directly Exploitable**
Impact	**High**
Confidentiality	An attacker with access to the database instances or EBS volumes can read the data directly without cracking the encryption first.
Integrity	There is no direct impact to data integrity.
Availability	There is no direct impact to availability.
Exposure	This vulnerability exposes the data within database instances and EBS Volumes. The data stored could aid an attacker in further attacks on the system.
Affected Users	All users that rely on the functionality of the AWS Account are affected.
Likelihood	**Low**
Skill Required	There is significant skill required in order to gain access to the service account and read the contents of the databases or EBS volumes.
Conditions and Complexity	An attacker would need access to the AWS accounts in order to gain access to the RDS and Redshift instances and EBS Volumes.
Discoverability	Once access to the service account is achieved, this is relatively easy to discover.
Reproducibility	This is always reproducible.

Background Information

AWS EBS provides persistent block storage for EC2 instances by abstracting the underlying physical hardware. EBS volumes can be formatted to use any file system, some of which may already support encryption but require the users to manage their own keys. As an alternative, AWS provides a transparent mechanism for encrypting EBS volumes using the AWS KMS

system. EBS encryption prevents attackers with physical or internal access to the AWS system from reading or modifying the data stored on the EBS volumes.

Amazon RDS and Redshift encrypted DB instances provide an additional layer of data protection by securing data from unauthorized access to the underlying storage. Users can use Amazon RDS encryption to increase data protection of applications deployed in the cloud, and to fulfill compliance requirements for data-at-rest encryption.

Problem Details

The following AWS accounts were evaluated for this configuration review:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

Unencrypted EBS Volumes

Some EBS volumes in the AWS account used by MetaCorp were found to not have encryption enabled via KMS. Usually, this will be an issue, however, MetaCorp may have disk encryption like dm-crypt enabled for these volumes. If these volumes are encrypted by dm-crypt or a similar disk encryption subsystem, then this issue becomes a problem of auditability rather than security.

The following accounts are affected:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)

240 of *551* of EBS volumes for *78927769 (Cloud Platform)* were found to not have encryption enabled via KMS.

Of the *240* unencrypted volumes, *205* have no *Name* tag fields.

Here is a breakdown for the other *311* named volumes, by their prefix (e.g. Prefix of "we" for "we-4-prod "):

Name Prefix	Number of Volumes With the Prefix
cows	241
application	10
common	10
common_production	8
common_production_cows	8
cfs_production	8
cfs_production_cows	8
cfs_production_angus	6
cfs_production_holstein	6
cfs_production_jersey	6

Unencrypted RDS Instance Storage

Some RDS instances do not have encryption enabled. These instances may contain sensitive data including customer data. Most of the findings were related to the DEV accounts, so it is likely that there is no customer data in these RDS instances. The following accounts are affected:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

Account ID	Region	Name
40190371	us-east-2	x3PHR4BZ1R-0
40190371	us-east-2	x3PHR4BZ1R-1
36584556	us-west-2	x3PHR4BZ1R-0
36584556	us-west-2	x3PHR4BZ1R
36584556	us-west-2	UKgKXV9pa-us-west-2a
78927769	us-east-1	vekTnKIKP8-us-east-1

Disabled Redshift Cluster Database Encryption

Some Redshift clusters do not have encryption enabled.

The following accounts are affected:

- 40190371 - METACORPAWS (Master Account)

Account ID	Region	Name
40190371	us-east-2b	GnKBZrdMiD2qp-prod
40190371	us-west-2b	GnKBZrdMiD2qp-test

Test Steps

Test Configuration

The following is needed in order to reproduce this issue:

- AWS CLI configured for the desired account
  - http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html
- Scout2 AWS security auditing tool
  - https://github.com/nccgroup/Scout2

Steps to Reproduce

1. Run the following Scout2 command from the command line, which will scan all AWS services in the account:

```
Scout2
```

1. Open the HTML report generated by Scout2.

2. In the navigation bar, navigate to "Computer/EC2/Dashboard".
3. In the Dashboard, click the "EBS volume not encrypted" link for a list of all unencrypted EBS volumes.
4. In the navigation bar, navigate to "Database/RDS/Dashboard".
5. In the Dashboard, click the "Instance storage not encrypted" link for a list of all unencrypted RDS instances.
6. In the navigation bar, navigate to "Database/Redshift/Dashboard".
7. In the Dashboard, click the "Cluster database encryption disabled" link for a list of all unencrypted Redshift clusters.

Remediation

Ensure sensitive EBS volumes are encrypted. Review all unencrypted EBS volumes and ensure that, if they store sensitive data, they are either encrypted using EBS encryption or through other at-rest data encryption mechanisms such as dm-crypt, eCryptfs, or BitLocker. Enable encryption on all RDS instances. Encryption should be enabled for all RDS instances that store sensitive customer data. More information can be found below.

- https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html

Enable encryption on all Redshift clusters. Encryption should be enabled for all Redshift clusters that store sensitive customer data. More information can be found below.

- https://docs.aws.amazon.com/cows/latest/developerguide/services-redshift.html

Additional Resources

- Amazon EBS Encryption
  - http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html
- AWS Key Management Service Best Practices
  - https://d0.awsstatic.com/whitepapers/aws-cows-best-practices.pdf (PDF Link)

Problem Report 2 - AWS Account Misconfiguration - Logging Disabled for Multiple Services

Logging is disabled for ELB, Redshift, S3, and VPC subnets. As a result, malicious activities performed by an attacker might go untracked and unnoticed.

Component	AWS Account
STRIDE	Repudiation
CWE	CWE-778: Insufficient Logging
CVSS v2 Score	0.0 (AV:N/AC:H/Au:S/C:N/I:N/A:N)
CVSS v3 Score	0.0 CVSS:3.0/AV:N/AC:H/PR:L/UI:N/S:U/C:N/I:N/A:N

Overall Severity	**Minimal**
Vulnerability Type	**Defense in Depth**
Impact	**Low**
Confidentiality	Data confidentiality is not directly affected by this finding.
Integrity	There is no impact on data integrity; malicious activities however might not be tracked with verifiable integrity.
Availability	There is no impact on service or data availability.
Exposure	There is no data loss associated with this finding.
Affected Users	All users of the service are affected.
Likelihood	**Low**
Skill Required	Very little skill is required to exploit this vulnerability once an attacker has access to perform the actions that are not being logged. However, gaining this access may be difficult.
Conditions and Complexity	An attacker must have access to perform actions on this account, once they have this they are able to perform actions which are not logged.
Discoverability	With access to the log storage, this is easy to discover. Gaining access, however, may be difficult.
Reproducibility	This is always reproducible due to it being a configuration state.

Background Information

In general, access logs, activity logs, and network flow logs are useful for detecting unusual activity, or determining the cause of a security incident.

Access logs for ELB captures and logs all requests, including requests that never make it to back-end services. This is helpful for detecting malformed requests, which is a sign of an attempted attack.

User activity logs in Redshift will log every query performed on the database. Access logging for S3 buckets logs details about every request, including the requester, the bucket which the request was sent to, any objects that may be a target of an action, and the type of action.

Flow logs for VPC subnets enable the logging of network traffic, useful for traffic analysis in response to incidents, or for use in early detection of a network breach.

Problem Details

The following AWS accounts were evaluated for this configuration review:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

Lack of Access Logs in ELB

Access logs in load balancers allow traffic analysis in response to security incidents and enable the identification of network security issues. The following accounts are affected:

- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

78927769 - COWSCLOUD (Cloud Platform)

12 out of *12* ELB load balancers do not have access logs enabled.

36584556 angus_feed (Business Intelligence)

5 out of *5* ELB load balancers do not have access logs enabled.

Lack of Access Logs in ELBv2

Access logs in load balancers allow traffic analysis in response to security incidents and enable the identification of network security issues. The following accounts are affected:

- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

Account ID	Name	Availability Zones
78927769	cows-prod-elb	us-east-2b,us-east-2c
36584556	cows--test-elb	us-west-2b,us-west-2c

User Activity Logging Disabled in Redshift Parameter Groups

Some Redshift parameter groups have user activity logging disabled. As a result, malicious activity occurring on the Redshift clusters may be easily repudiated or be left undiscovered. The following accounts are affected:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

Account ID	Group Name	Region
40190371	cows.redshift-1.0	us-east-2
40190371	cows.redshift-1.0	us-west-2
78927769	cows.redshift-1.0	us-east-1

S3 Bucket Access Logging Disabled

All S3 buckets for all accounts have logging disabled, except for four belonging to 36584556 (angus_feed). This is a concern, especially when some S3 buckets are accessible to the public, some of which may have been left open to access unintentionally. Without access logging, there is no way to determine whether unusual activity has taken place on any of the affected S3 buckets. The following accounts are affected:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)

The following table shows S3 buckets that have access logging **enabled**.

Account ID	Bucket Name	Region
36584556	static.archive.com	us-east-1
36584556	com.metacorp.context.archive	us-west-1
36584556	com.metacorp.archive.us-east-1-clean	us-east-1
36584556	com.metacorp.archive.us-east-1	us-east-1

For the above buckets, access logging makes sense, especially for the archives. The archive buckets are not accessible to the public, but it is still prudent to enable access logging for these, in case somebody with malicious intent manages to somehow gain access.

VPC Subnet Without Flow Logs

All VPC subnets were found to not have this feature enabled. The following accounts are affected:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

Test Steps

Test Configuration

The following is needed to reproduce this issue:

- AWS CLI configured for the desired account
  - http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html
- Scout2 AWS security auditing tool
  - https://github.com/nccgroup/Scout2

Steps to Reproduce

1. Run the following Scout2 command from the command line, which will scan all AWS services in the account:

```
python Scout2.py
```

2. Open the HTML report generated by Scout2
3. In the navigation bar, navigate to "Compute/Elb/Dashboard"
4. Select "Lack of access logs" on the Dashboard to see a list of ELB load balancers without access logs.
5. In the navigation bar, navigate to "Compute/Elbv2/Dashboard"
6. Select "Lack of access logs" on the Dashboard to see a list of ELBv2 load balancers without access logs.
7. In the navigation bar, navigate to "Database/Redshift/Dashboard"
8. Select "User activity logging disabled" on the Dashboard to see a list of Redshift parameter groups with activity logging disabled.
9. In the navigation bar, navigate to "Database/S3/Dashboard"
10. Select "Bucket access logging disabled" on the Dashboard to see a list of S3 buckets with activity logging disabled.

Remediation

Enable access logging for ELB.

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

Enable audit logging for Redshift databases.

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

Enable access logging for S3 buckets.

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

Problem Report 3 - AWS Users Without Multi-Factor Authentication

MetaCorp's AWS account has active Identity and Access Management (IAM) users with password access that do not have multi-factor authentication (MFA) enabled. An attacker that can compromise or brute force an AWS account user's password can log into the AWS account and carry out actions as that user.

Component	AWS Account
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Elevation of Privilege
CWE	CWE-308: Use of Single-factor Authentication
CAPEC	CAPEC-49: Password Brute Forcing
CVSS v2 Score	5.1 (AV:N/AC:H/Au:N/C:P/I:P/A:P)
CVSS v3 Score	5.6 CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L

Overall Severity	**Medium**
Vulnerability Type	**Defense in Depth**
Impact	**High**
Confidentiality	Confidentiality is not directly affected by this issue.
Integrity	Integrity is not directly affected by this issue.
Availability	Availability is not directly affected by this issue.
Exposure	An attacker that can compromise or brute force an AWS account user's password can log into the AWS account and carry out actions as that user.
Affected Users	All users relying on the AWS account are affected.
Likelihood	**Low**
Skill Required	Using a compromised password requires a trivial amount of skill. However, it would require additional skill to identify other vulnerabilities in order to steal or brute force a password.
Conditions and Complexity	An attacker would have to brute force or capture a password through other vulnerabilities in order to exploit this issue.
Discoverability	This issue is easy to identify if an attacker knows a user's username or email address.
Reproducibility	This issue is 100% reproducible.

Background Information

AWS accounts can be accessed using usernames and passwords or IAM Access Keys, but AWS Console access requires a username and password. Password strength requirements can be customized by an AWS account administrator and can potentially be fairly weak. MFA systems improve the security posture of the application by increasing the number of items a user must provide as evidence that they are who they claim to be. Generally this is done with a combination of something they know, something they have, or something they are. A common multi-factor authentication scheme is to utilize a password (something they know) and one-

time use token generated by, or sent to, a mobile device (something they have). Requiring MFA for AWS Users with password access is an important defense in depth security control that increases the difficulty of compromising an IAM user account.

Problem Details

Some IAM user accounts with password access were found to not have MFA enabled.

The following AWS accounts were evaluated for this configuration review:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

Affected Areas

Account ID	User Name
40190371	AmayahNicholson
78927769	SameeraBurton
78927769	MuneebLovell
78927769	GurpreetNash
36584556	BrandyTyson
36584556	MaisieEstes
36584556	LeenaDiaz
36584556	ByronONeill
36584556	ZeeshanMelton
36584556	IrvingGillespie
36584556	TaylaCresswell
36584556	VictoriaLuna
84751486	CalistaCouch
84751486	MelodyHawes
84751486	FarrahDorsey
84751486	YazminWagstaff
84751486	SorenHatfield
84751486	VictorHobbs
84751486	FranciszekRosa
84751486	HafsahGaines
84751486	AedanMason

This list may not include all user accounts with MFA disabled, as there may be additional accounts not examined. It is suggested that, upon remediation, the development team audit all AWS accounts in order to find and fix related issues.

Test Steps

Test Configuration

The following is needed in order to reproduce this issue:

- AWS CLI configured for the desired account
  - http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html
- Scout2 AWS security auditing tool
  - https://github.com/nccgroup/Scout2/

Steps to Reproduce

1. Run the following Scout2 command from the command line, which will scan all AWS services in the account:

   ```
   python Scout2.py
   ```

2. Open the HTML report generated by Scout2.

3. In the navigation bar, navigate to "Security/IAM/Dashboard".

4. Select "User without MFA" on the Dashboard to see a list of users without MFA enabled.

Remediation

Enable MFA for all IAM users with password access. Enforce MFA for all users with password access to the AWS account.

Review and remove unnecessary users. Ensure that there are no unnecessary users or roles in AWS accounts to reduce the attack surface.

Create policies requiring that users use MFA. For increased security, it is possible to create policies that require that a user has authenticated using MFA before performing any actions.

Additional Resources

- Using Multi-Factor Authentication (MFA) in AWS
  - http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa.html
- Tutorial: Enable Your Users to Configure Their Own Credentials and MFA Settings
  - http://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_users-self-manage-mfa-and-creds.html
- Requiring Multi-Factor Authentication (MFA) in AWS
  - http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_configure-api-require.html

Problem Report 4 - AWS Insecure Network ACLs

Network Access Control Lists (ACLs) in the AWS account are configured with insecure network traffic rules. Network ACLs serve as an important defense in depth security control in the event insecure Security Groups are used. Insecure Network ACLs increase the attack surface of EC2 instances within a Subnet by potentially exposing ports and services to external attackers. In addition, in the event of a compromise, secure Network ACLs can prevent the attacker from pivoting within the Subnet.[0]_

Component	AWS Account
STRIDE	Elevation of Privilege
CWE	CWE-183: Permissive Whitelist
CAPEC	CAPEC-300: Port Scanning
CVSS v2 Score	2.6 (AV:N/AC:H/Au:N/C:P/I:N/A:N)
CVSS v3 Score	3.7 CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

Overall Severity	**Medium**
Vulnerability Type	**Defense in Depth**
Impact	**Medium**
Confidentiality	Information about what services are running on EC2 instances within a Subnet is exposed.
Integrity	Integrity is not directly affected by this issue.
Availability	Availability is not directly affected by this issue.
Exposure	Network ACLs serve as an important defense in depth security control in the event insecure Security Groups are used. Insecure Network ACLs increase the attack surface of EC2 instances within a Subnet by potentially exposing ports and services to external attackers, as well as potentially allowing compromised EC2 instances to perform outbound communication.
Affected Users	All users relying on the AWS account are affected.
Likelihood	**Medium**
Skill Required	Identifying open ports is easily performed by an attacker with off-the-shelf tools. Exploiting exposed ports requires a more skilled attacker.
Conditions and Complexity	Identifying open ports is trivial, however, this issue also requires that insecure Security Groups be attached to EC2 instances within the Subnet.
Discoverability	This issue is easy to discover through off-the-shelf port scanning tools.
Reproducibility	This issue is 100% reproducible.

Background Information

AWS Network ACLs act as firewalls that can be attached to AWS Subnets to control ingress and egress traffic within a VPC. They serve as an additional layer of security in the event that EC2

Security Groups are misconfigured. By default, every VPC is created with a default Network ACL that allows all inbound and outbound traffic, and Subnets within the VPC will have this insecure Network ACL applied to them. Network ACLs should follow the Principle of Least Privilege and only whitelist traffic that is necessary for the function of the instances within the Subnet.

Problem Details

Some Network ACLs in the AWS account used by MetaCorp were found to be insecurely configured.

The following AWS accounts were evaluated for this configuration review:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

78927769 (COWSCLOUD)

All Network ACLs were found to allow all ports, for both ingress and egress traffic. As a result, all Subnets for these accounts have the same ACL insecure rules.

There are *22* Network ACLs on this account, with *48* Subnets associated with those ACLs. The following table is a partial list of those Network ACLs.

Account ID	Region	NACL	Direction	Protocol	Port Range	# Associated Subnets
78927769	ap-south- 1	acl-4vwS2B3q3M	Both	ALL	0-65535	24
78927769	eu-north-1	acl-DXD7VJsHEP	Both	ALL	0-65535	24

(20 entries omitted from table)

40190371 (METACORPAWS)

All Network ACLs were found to allow all ports, for both ingress and egress traffic. As a result, all Subnets for these accounts have the same ACL insecure rules.

There are *4* Network ACLs on this account, with *40* Subnets associated with those ACLs.

Account ID	Region	NACL	Direction	Protocol	Port Range	# Associated Subnets
40190371	ap-south-1	acl-TPr835aJzz	Both	ALL	0-65535	10
40190371	eu-north-1	acl-EX7vP1jQb	Both	ALL	0-65535	10

Account ID	Region	NACL	Direction	Protocol	Port Range	# Associated Subnets
40190371	ap-northeast-1	acl-u2wrpGnKBZ	Both	ALL	0-65535	10
40190371	ap-southeast-2	acl-1jQbxyiduP	Both	ALL	0-65535	10

36584556 (angus_feed)

All Network ACLs were found to allow all ports, for both ingress and egress traffic. As a result, all Subnets for these accounts have the same ACL insecure rules.

There are *5* Network ACLs on this account, with *25* Subnets associated with those ACLs.

Account ID	Region	NACL	Direction	Protocol	Port Range	# Associated Subnets
36584556	ap-south-1	acl-ViOEPFyG	Both	ALL	0-65535	5
36584556	eu-north-1	acl-U79EwzZ7	Both	ALL	0-65535	5
36584556	ap-northeast- 1	acl-3Wwldvep	Both	ALL	0-65535	5
36584556	ap-southeast- 2	acl-nKtteKhW	Both	ALL	0-65535	5
36584556	eu-west-1	acl-GYHdhix6	Both	ALL	0-65535	5

This list may not include all insecurely configured Network ACLs, as there may be additional accounts not examined. It is suggested that upon remediation, the development team audit all AWS accounts in order to find and fix related issues.

Test Steps

Test Configuration

The following is needed in order to reproduce this issue:

- AWS CLI configured for the desired account
  - http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html
- Scout2 AWS security auditing tool
  - https://github.com/nccgroup/Scout2/

Steps to Reproduce

1. Run the following Scout2 command from the command line, which will scan all AWS services in the account:

```
Scout2.py
```

2. Open the HTML report generated by Scout2.
3. In the navigation bar, navigate to "Network/VPC/Dashboard".
4. Note the highlighted issues.

Remediation

Ensure all Network ACLs have secure permissions. Review all used Network ACLs, including the default networks ACLs, and ensure they follow the Principle of Least Privilege. By default, deny access to all and only allow access to allow specific network traffic necessary for the operation of the system.

Problem Report 5 - IAM Policies With Permissive AssumeRole

Identity and Access Management (IAM) Policies in the AWS account have permissive `sts:AssumeRole` rules. If an IAM User has a lax IAM Policy attached to them, they may be able to make an AssumeRole call to retrieve access credentials for an IAM Role with higher AWS privileges than they currently have.

Component	AWS Account
STRIDE	Spoofing, Elevation of Privilege
CWE	CWE-183: Permissive Whitelist
CAPEC	CAPEC-1: Accessing Functionality Not Properly Constrained by ACLs
CVSS v2 Score	9.3 (AV:N/AC:M/Au:N/C:C/I:C/A:C)
CVSS v3 Score	9.8 CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
OWASP Reference	OWASP Top 10 2013-A7: Missing Function Level Access Control

Overall Severity	**Critical**
Vulnerability Type	**Directly Exploitable**
Impact	**High**
Confidentiality	All data on the accounts may be compromised.
Integrity	All data on the accounts may be compromised.
Availability	All data on the accounts may be compromised.
Exposure	The account is fully exposed.
Affected Users	All users relying on the AWS account are affected.
Likelihood	**High**
Skill Required	Moderate understanding of AWS and minor understanding of MetaCorp's implementation of AWS roles is required for this attack.
Conditions and Complexity	An attacker can do perform this attack anonymously, with no additional complexity.
Discoverability	This issue is difficult to identify (for external attackers) due to requiring knowledge of the account's existing role policies.
Reproducibility	This issue is 100% reproducible.

Background Information

IAM allows AWS customers to define fine-grained access control to AWS resources and services using IAM Policies. IAM Policies can be attached to IAM Users, Groups, or Roles and are made up of explicit deny or allow access rules. Using IAM Policies, permissions can be defined using the principle of least privilege. However, due to the granular nature and complexity of IAM Policies, it is possible to erroneously create insecure IAM Policies that provide more privileges to attached entities than intended.

Security Token Service (STS) allows users to make API calls to request temporary access

credentials for IAM Users or IAM Roles. When AWS users are given permissions to call STS AssumeRole, they are given temporary IAM Access Keys for that IAM Role. This functionality is often used to provide cross-account access or other federated access. If an AssumeRole rule in an IAM Policy is overly permissive, the attached entity could assume IAM Roles with the appropriate Trust Relationship, which may have higher privileges than intended for the IAM User.

For example, if an IAM User has the following IAM Policy attached to it:

IAM Policy

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

The User would be able to assume the privileges of any IAM Role they fulfill the Trust Relationship for. This is especially important to consider, as there may be Trust Relationships established in other AWS accounts. These Trust Relationships would be difficult to audit.

Alternatively, if an account contains the following IAM Policy:

IAM Policy

```
{
 "Version": "2012-10-17",
 "Statement": [
     {
         "Effect": "Allow",
         "Action": "sts:AssumeRole",
         "Resource": "*",
         "Principal": {
             "AWS": "*"
          }
     }
 ]
}
```

The lack of specification on the Principal field would allow any user from any AWS account to assume the role and enter the account, inheriting any other permissions that the role grants. In this particular case, Resource: * would allow the cross-account AssumeRole to escalate to a complete account takeover.

Problem Details

Some IAM Policies in the AWS accounts used by MetaCorp were found to have overly permissive `sts:AssumeRole` rules.

In particular, certain policies were found that specified a `*` for the Principal of the role

assumption. This is dangerous, as it allows for users from other accounts to act as the principal and assume high-powered, administrator roles within the MetaCorp AWS accounts.

Other policies were found that allowed for `*` for the Resource of the role assumption. Doing so allows a user with this role to assume more powerful roles that give additional permissions. The various roles and policies that contained these vulnerabilities are enumerated below.

The following AWS accounts were evaluated for this configuration review:

- 40190371 - METACORPAWS (Master Account)
- 78927769 - COWSCLOUD (Cloud Platform)
- 36584556 - angus_feed (Business Intelligence)

Affected Areas

All three AWS accounts evaluated had the following three roles, which allowed for cross-account role assumption:

- ReadOnly
- PowerUser
- Admin

In addition to the above, the following table presents a list of accounts' policies that allowed for privilege escalation via assuming more powerful roles:

Account ID	Policy Name
78927769	UserAccess_Secure
36584556	UserAccess_Insecure
84751486	Admin_Verified
84751486	OneOffAppAccess
36584556	TempAccess

This list may not include all insecurely configured IAM Policies, as there may be additional accounts not examined. It is suggested that, upon remediation, the development team audit all AWS accounts in order to find and fix related issues.

Test Steps

Test Configuration

The reproduction shown will demonstrate the more dangerous of the two misconfigurations (account takeover). For this reproduction, the following will be needed:

- An attacker-owned (non-MetaCorp) AWS account
  - Create API credentials for this user and provide them to the AWS CLI
  - This reproduction will refer to a user named jdean-fullaccess
- A device capable of registration for Multi-Factor Authentication (MFA), such as a smart phone
- This reproduction will use a smart phone with the free Google Authenticator app

installed

Steps to Reproduce

Note: For the purposes of this reproduction, the `Admin` role belonging to the `78927769` AWS account will be shown. The same role may be assumed in any of the accounts to fully compromise them. The `UserAccess_Secure` role may also be used to the same effect, with the additional step of needing to call `AssumeRole` an extra time (to escalate from `User` to `Admin`).

1. With the attacker-controlled AWS credentials within the CLI, create a virtual MFA device:

```
aws iam create-virtual-mfa-device --virtual-mfa-device-name mfatest --outfile scanfile.png
--bootstrap-method QRCodePNG
```

In the above text, note that `mfatest` is a user-chosen name. This is simply the name used in this example. The output file, `scanfile.png`, is the file that will be used to supply the MFA seed to the smart phone app.

2. The file, `scanfile.png` will be created. Open this file in any image viewer, and use the Google Authenticator app to add this QR Code as a new MFA device.
3. Now that the MFA seed is stored within the phone application, retrieve two MFA tokens in a row and place these into the following API call:

```
aws iam enable-mfa-device --user-name jdean-fullaccess --serial-number
arn:aws:iam::ACCOUNT_NUMBER:mfa/mfatest --authentication-code-1 token1 --
authentication-code-2 token2
```

In the above:

- `jdean-fullaccess` is the user who owns the API credentials making the call
- `ACCOUNT_NUMBER` is the account number of the attacker-controlled AWS account, not MetaCorp's
- `mfatest` is the name of the MFA device created in Step (1) above
- `token1` and token2 are two consecutive MFA tokens provided by the Google Authenticator app

4. The attacker account may now call `sts:AssumeRole` against the MetaCorp account, supplying its own attacker-created MFA credentials to allow it access into the account as follows:

```
aws sts assume-role --role-arn arn:aws:iam::78927769:role/Admin_Role_if_MFA --
role-session-name anything --serial-number arn:aws:iam::ACCOUNT_NUMBER:mfa/mfatest -
-token-code token
```

In the above:

- `anything` is the session name used for logging/collision avoidance. It cannot be null, but

is up to the user. We used `pentesting`

- `ACCOUNT_NUMBER` is the account number of the attacker-controlled AWS account, not MetaCorp's
- `mfatest` is the name of the MFA device created in Step (1) above token is an MFA token for the user provided by Google Authenticator

5. If all steps before now were done properly, the server will respond with the following information:

```
{
   "AssumedRoleUser": {
         "AssumedRoleId": "AW2INMUDAFI:pentesting",
         "Arn": "arn:aws:sts::78927769:assumed-role/Admin_Verified/pentesting"
   },
   "Credentials": {
      "AccessKeyId": "SAVE_ACCESS_KEY",
      "SessionToken": "SAVE_SESSION_TOKEN",
      "Expiration": "2019-05-02T01:21:14Z",
      "SecretAccessKey": "SAVE_SECRET_KEY"
   }
}
```

In the above:

- `pentesting` was the session name provided above, marked as `anything` in the previous step
- `SAVE_ACCESS_KEY` and the other two variants will be used in the below step. Take note of them.

6. Set the access key, secret key, and session token obtained in the last step as environmental variables within the CLI. This is done via "export" on Linux, or "set" on Windows. The below functionality will be for Linux.

```
export AWS_ACCESS_KEY_ID=SAVE_ACCESS_KEY
export AWS_SECRET_ACCESS_KEY=SAVE_SECRET_KEY
export AWS_SESSION_TOKEN=SAVE_SESSION_TOKEN
```

7. Finally, call the CLI using the `sts:GetCallerIdentity` functionality to determine what role has been assumed in the system:

```
aws sts get-caller-identity
```

This is similar to calling the Linux `whoami`, and the server will respond as shown below:

```
{
   "Account": "78927769",
   "UserId": "AW2INMUDAFI:pentesting",
   "Arn": "arn:aws:sts::78927769:assumed-role/Admin_Verified/pentesting"
}
```

Since Account ID 78927769 is one of the MetaCorp AWS accounts, this is a sufficient proof of concept that the account takeover was successful.

Remediation

Restrict AssumeRole to specific IAM Roles in IAM Policies. Instead of using `*` or other overly permissive values for the `Resource` and `Principal` Policy element, limit the assuming Principals to the owner's account (via `arn:aws:iam::############:*`) and limit the roles that may be assumed only to the necessary roles. A sample is shown below:

Secure IAM Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow", "Action": "sts:AssumeRole",
         "Resource": "arn:aws:iam::123456789:role/User_Verified",
         "Principal": {
            "AWS": "arn:aws:iam::123456789:* "
      }
   }
]
}
```

Additional Resources

- Granting a User Permissions to Switch Roles
  - http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html
- AssumeRole
  - http://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html
- IAM Policies
  - http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html
- Help with creating/assigning a Virtual MFA Device
  - https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_mfa_enable_cliapi.html

Observations

Over the course of testing, there were a total of 5 problems identified. In addition, there were several other observations made about MetaCorp's AWS Configuration. While not severe enough to warrant problem reports, Security Innovation recommends that MetaCorp investigate these in addition to the security-related issues that were found.

Observation 1 - EC2 Security Groups Have SSH Port Open To All

Some EC2 security groups have port 22 open for inbound traffic. The security groups should be reviewed to determine whether it is acceptable to leave this port open. In the event that any ports do not need to be open, they should be closed from public access.

The following is a list of security groups that have an inbound rule that opens port 22 for any source:

Account ID	ID	Name	Region
78927769	sg-45264992	metacorp-sso	us-east-1
78927769	sg-30016491	cows-zFHVBgl2T4nEpY7yCXyzZxACv	us-east-1
78927769	sg-38270622	cows-RLzPl2wOTPr835aJzzOFwSUBH	us-east-1
78927769	sg-47351190	cows-MzyBP3ueLJ21AZ4vwS2B3q3Mb	us-east-1
78927769	sg-66266222	cows-hU2a6vScvcn107gCoYruFj3zX	us-east-1

Observation 2 - RDS Auto Minor Version Upgrade Disabled

Automatic minor version upgrade is disabled for some RDS instances. This is a feature that upgrades instances when a new minor version is available for their respective database engines. Enabling this feature is recommended.

The following is a list of RDS instances that have automatic minor version upgrade disabled:

Account ID	Name	Region
84751486	cows-us-east-1	us-east-1
84751486	cows-us-east-1	us-east-1
84751486	cows-us-east- 1	us-east-1
84751486	cows-us-east-1- replica-1	us-east-1
84751486	cows-us-east-1	us-east-1
78927769	cows-partner	us-east-1
78927769	cows-platform	us-east-1
78927769	cows-common-platform	us-east-1

Observation 3 - RDS Short Backup Retention Period

Some of the RDS instances have their backup retention period set to 14 days or less. These RDS instances should be reviewed to determine whether they should have longer retention periods.

Account ID	Name	Region	Retention Period (days)
78927769	cows-partner	us-east-1	7
78927769	cows-ii	us-east-1	7
78927769	cows-common-ae	us-east-1	7
78927769	cows-platform	us-east-1	7
78927769	cows-dev	us-east-1	7
78927769	cows-dev	us-east-1	7
78927769	cows	us-east-1	7
78927769	cows-common-ee	us-east-1	7
78927769	cows-platform-eia	us-east-1	7
78927769	cows-events	us-east-1	7

Tools

While performing the security configuration review, the following tools were employed:

Tool	Description	Link
Scout2	Security auditing tool for AWS environments	https://nccgroup.github.io/Scout2/
AWS Command Line Interface	A command line tool for managing AWS services.	https://aws.amazon.com/cli/

Recommended Next Steps

This section contains our recommendations for areas that may benefit from additional testing. For each section, we describe why it is important to test these sections, either more thoroughly or for the first time.

Retest After Remediation - A retest of the MetaCorp accounts is recommended to be performed when the problems found as a result of this test have been remediated. This validates the remediation put in place and ensures that other vulnerabilities have not been introduced in the course of remediation.

Other AWS Accounts - In early scoping discussions, other AWS accounts were also mentioned as less-important, staging or development testing environments. It is recommended that these accounts be reviewed at least for high-severity issues like complete account takeovers by anonymous users.

Follow-up

Inconsistencies, errors, and reproducibility problems associated with this report should be directed through the customer contact person to the tester and preparer indicated at the beginning of this report.