CLIENTNAME

CLOUD PENETRATION TEST

V1.0

March 2, 2023

TrustFoundry

TABLE OF CONTENTS

SUMMARY OF WORK PERFORMED

A cloud configuration security assessment was performed on the ClientName AWS environment. All AWS services for the ClientName organization were in scope for this assessment. Read-only access was provisioned to the <name> AWS account. The following AWS account ID was used for testing:

- 111111111111

The following regions were in scope for testing:

- us-east-1
- us-east-2

A network penetration test was performed on the ClientName AWS internal cloud environment. TrustFoundry was provisioned with access to the following EC2 instance:

- <example ARN>

TrustFoundry could attempt to pivot from this host to gain access to other hosts and data in the cloud environment, with the end goal of obtaining access to the following host:

- <example ARN>

The assessment was completed during the following dates:

Date	Task
January 1, 2023	Testing Started
January 5, 2023	Testing Completed
January 9, 2022	Report Delivered

Methodology

Cloud Security Assessment Methodology

The cloud security assessment uses read-only access to cloud environments to ascertain the security of the configuration and architecture of cloud resources in the organization. A primary goal was to discover resources that were unintentionally exposed to the public. Security controls were evaluated to discover lax policies that would enable adversaries to escalate privileges or perform unauthorized actions. Additionally, best practices are recommended that could be optionally enabled to further improve the security of the ClientName cloud infrastructure.

A cloud security assessment is conducted in accordance with best practices and checklists developed over years of testing to ensure thorough coverage. TrustFoundry's cloud security assessment consists of the following phases:

- **Pre-Assessment** – TrustFoundry will request access to the systems in advance of the test and guide the customer through the testing process on a pre-assessment call.
- **Target Profiling** – TrustFoundry will determine all security controls used in the environment.
- **Vulnerability Discovery** – TrustFoundry will examine security controls to identify vulnerabilities or areas within those controls that could be improved.
- **Vulnerability Validation** – All identified vulnerabilities will be confirmed where possible to improve the accuracy of the report, and to provide simple reproduction steps where possible.
- **Reporting** – A report outlining all identified issues will be prepared which focuses on presenting identified vulnerabilities in a manner which makes them effective to remediate.

In addition to a variety of open-source tools, exploits, and utilities used on an as-needed basis, the following tools may be used when conducting a cloud security assessment:

- Aaia
- AWS CLI
- AWS PWN
- aws-recon
- AWS-Security-Toolbox (AST)
- CCAT
- CloudBrute
- Cloudfrunt
- Cloudjack
- Cloud-reports
- Cloudsplaining
- CloudSploit Scans
- cs-suite
- Dufflebag
- DumpsterDiver
- GitLeaks
- GreyhatWarfare
- LambdaGuard
- Lambda-Proxy
- Nimbostratus
- Pacu
- Prowler
- ScoutSuite
- SkyArk
- TrailScraper
- TruffleHog
- WeirdAAL
- whispers
- Zeus

TrustFoundry

Executive Summary

Overall, ClientName's attack surface was found to be relatively minimal. The following positive practices were observed:

- Multi-Factor Authentication (MFA) was properly enforced across all accounts.
- Logging was correctly configured across all resources.

Five issues were identified during the cloud security assessment, one of which was high severity.

The high severity flaw occurs because sensitive information is stored in instance metadata. An attacker who gains access to this metadata could potentially use the sensitive information to pivot to other hosts or escalate their privileges in the environment.

The medium severity finding is due to the lack of a password policy, allowing AWS users to choose very short, weak passwords.

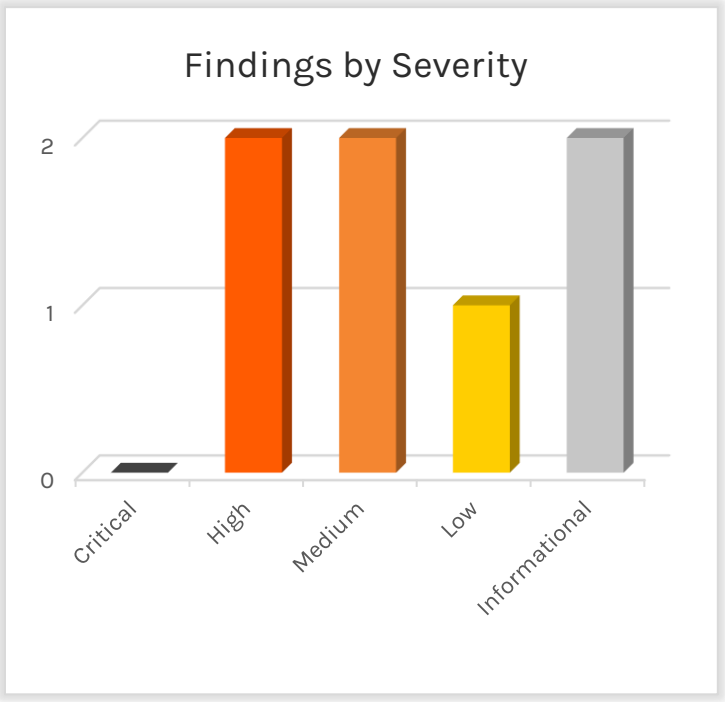
The low severity issue was due to a lack of key rotation for six IAM accounts. One set of unused credentials was not disabled, which was an informational severity issue. Finally, a permissive network policy (security group) was identified but was not observed to be in use and the associated finding was assigned informational severity.

Two vulnerabilities were identified during the cloud network penetration test. The first was a high severity vulnerability related to a MongoDB instance that allowed unauthenticated access. An attacker could view any data stored in the database and potentially leverage the data to take over user accounts.

One medium severity flaw related to the use of a self-signed certificate was identified. This can make attacks related to traffic interception more difficult for users to identify.

The following chart shows the distribution of findings across all severity levels:

Severity	Count of Findings
Critical	0
High	2
Medium	2
Low	1
Informational	2



SUMMARY OF FINDINGS

Finding information in this report is presented in order of severity. Severity ratings are presented without the knowledge of the business risk that the vulnerabilities present.

The following vulnerabilities were discovered during the cloud configuration security assessment:

#	Finding	Severity
1	Sensitive Information in Instance Metadata	High
2	Weak IAM Password Policy	Medium
3	Lack of Key Rotation	Low
4	Permissive Network Policy	Informational
5	Unused Credentials Not Disabled	Informational

The following vulnerabilities were discovered during the cloud network penetration test:

#	Finding	Severity
6	MongoDB Service Does Not Require Authentication	High
7	Self-Signed Certificate	Medium

FINDINGS AND RECOMMENDATIONS

Cloud Configuration Security Assessment

1. Sensitive Information in Instance Metadata

■ SEVERITY: HIGH

Finding Information

An EC2 instance was configured with user data within the metadata which includes sensitive information. An adversary with access to the EC2 instance can access this information in plain text from within the instance at the following URL:

- http://169.254.169.254/latest/user-data

Evidence

The output of the following command shows the base64-encoded string of the user metadata for EC2 instance i-111111111111.

```
aws ec2 describe-instance-attribute --instance-id i-111111111111 --attribute
userData --region us-east-1

{
    "InstanceId": "i-111111111111",
    "UserData": {
        "Value": "
IyEvYmluL2Jhc2gKVXNlckRhdGFDb25maWc6CmNfZGJfY29ubjogW3siY29ubl9zdHIiOiJEYXRhYmFzZ
T1wcm9kO1NlcnZlcj1wcm9kLmV4YW1wbGUuY29tO3VzZXI9ZXhhbXBsZTtwYXNzd29yZD1leGFtcGxlcG
Fzc3dvcmQ7Iiwid3JpdGFibGUiOnRydWV9XQpjX2xvZ19kYl9jb25uOiBbeyJjb25uX3N0ciI6IkRhdGF
iYXNlPXByb2Q7U2VydmVyPXByb2QtYW5hbHl0aWNzLmV4YW1wbGUuY29tO3VzZXI9ZXhhbXBsZTtwYXNz
d29yZD1leGFtcGxlMjsiLCJ3cml0YWJsZSI6dHJ1ZX1dCmV4YW1wbGU8cG93ZXJzaGVsbD4gLiJEOlxzY
3JpcHRzXGluc3RhbGxzY3JpcHQucHMxIiB8IE91dC1GaWxlICJEOlx0ZW1wXGluc3RhbGxhdGlvbi5sb2
ciOyA8L3Bvd2Vyc2hlbGw+Cg=="
    }
}
```

Decoding the string reveals the EC2 user metadata, including database passwords:

```
echo 'IyEvYmluL2Jhc2g<SNIP>2Vyc2hlbGw+Cg==' |base64 -d

#!/bin/bash
UserDataConfig:
c_db_conn:
[{"conn_str":"Database=prod;Server=prod.example.com;user=example;password=example
password;","writable":true}]
c_log_db_conn: [{"conn_str":"Database=prod;Server=prod-
analytics.example.com;user=example;password=example2;","writable":true}]
example<powershell> ."D:\scripts\installscript.ps1" | Out-File
"D:\temp\installation.log"; </powershell>
```

Affected Resources

- arn:aws:ec2:us-east-1: 111111111111:instance/i-111111111111

Impact

Metadata can only be accessed from within the instance itself, but the data is not protected by any cryptographic methods. Any adversary who can access the instance can view its metadata. Additionally, any software running on the instance could have access to the metadata API. A vulnerability in the software, such as Server-Side Request Forgery (SSRF), could give an adversary access to the sensitive information stored in the Metadata API.

In this case, access to this information would allow an adversary to access credentials to resources including the <example name> servers. This would allow an adversary to pivot from one compromised host to another.

Recommendations

Do not store sensitive data, such as passwords or keys, in the instance Metadata. Instead, use a configuration file or the Secrets Manager. Additionally, require Metadata v2.0 to add additional protections against SSRF attacks.

Additional Information

- CWE-256: Unprotected Storage of Credentials
- OWASP - Server Side Request Forgery
- AWS - Instance metadata and user data
- Add defense in depth against open firewalls, reverse proxies, and SSRF vulnerabilities with enhancements to the EC2 Instance Metadata Service

2. Weak IAM Password Policy

■ SEVERITY: MEDIUM

Finding Information

No password policy was found configured on the AWS account allowing users to set passwords that are short and/or lacking complexity.

Evidence

The AWS command `aws iam get-account-password-policy` returned a *NoSuchEntity* response indicating that the AWS account does not have an active IAM password policy. This can be observed in the following output:

```
$ aws iam get-account-password-policy --profile example

An error occurred (NoSuchEntity) when calling the GetAccountPasswordPolicy
operation: The Password Policy with domain name 111111111111 cannot be found.
```

Affected Locations

- This is a systemic issue on the entire AWS account (111111111111)

Impact

A weak password policy allows users to choose easily guessed passwords and keep them indefinitely. This issue's severity is partially mitigated as the root account was observed to have 2FA enabled.

Recommendations

Consider implementing a strong password policy to mitigate or prevent weak passwords.

The following are considered to be best practices for a password policy:

- Password Length: A minimum length of 14 characters is recommended.
- Minimum password age: This controls how long a user must wait before they can change their password.

This can be remediated with the following AWS CLI command:

```
aws iam update-account-password-policy --minimum-password-length 14
```

Additional Information

Refer to the following link for NIST best practices for password security guidelines:

- NIST Special Publication 800-63B

- Amazon - Setting an account password policy for IAM users

3. Lack of Key Rotation

■ SEVERITY: LOW

Finding Information

Keys were discovered that have not been changed for more than 90 days. AWS Keys should be treated like passwords and changed on a regular basis.

Evidence

The following data is from an AWS credential report generated on January 1, 2023 showing the active key has not been rotated for more than 90 days:

Field	Value
user	exampleuser
arn	arn:aws:iam::111111111111:user/exampleuser
user_creation_time	2018-02-20T13:13:03+00:00
access_key_1_active	TRUE
access_key_1_last_rotated	2022-03-04T12:03:50+00:00
access_key_1_last_used_date	2022-03-20T08:16:00+00:00
access_key_1_last_used_region	us-east-1
access_key_1_last_used_service	ecs

Affected Resources

- arn:aws:iam::111111111111:user/exampleuser

Impact

If keys are not rotated at regular, reasonably short intervals, an attacker who compromises a key will have a longer window of time to leverage the compromised key.

Recommendations

Rotate the affected credentials. Changing access keys on a regular schedule is a well-known security best practice because it shortens the period an access key is active and therefore reduces the attack window for compromised keys to be used.

Additional Information

- AWS Foundational Security Best Practices – IAM.3
- CWE-263: Password Aging with Long Expiration
- AWS - How to Rotate Access Keys for IAM Users

4. Permissive Network Policy

SEVERITY: INFORMATIONAL

Finding Information

A permissive network policy may allow attackers to access resources or services that should be protected by firewalls. Specifically, the following networking issues were discovered:

- Security Group opens port 22 from all

Evidence

The following configuration in AWS allows any IP to connect to SSH on port 22:

```
$ aws ec2 describe-security-groups --region us-east-1 --group-ids sg-111111111111
<example output>
```

Affected Resources

- arn:aws:ec2:us-east-1:111111111111:security-group/sg-111111111111

Impact

Any source IP address permitted in the policy could send attacks to these ports. This creates a wider attack surface for resources assigned to the policy. Automated scanners are constantly probing well-known ports and an adversary may use these scanners to determine possible targets.

This security group was not observed as being in use. As a result, the severity of this finding has been set to informational.

Recommendations

The principle of least privilege should be enforced to ensure externally accessible ports are reduced to the minimum needed to conduct business. If such services must be exposed, then a restriction on the source address could help to reduce the attack surface of the cloud infrastructure. Audit affected resources and determine if the most restrictive network ACL is in use.

Additional Information

- AWS Foundational Security Best Practices EC2.19
- CWE-284: Improper Access Control

5. Unused Credentials Not Disabled

SEVERITY: INFORMATIONAL

Finding Information

Credentials were discovered that have not been used for over 90 days. These credentials may be unnecessary.

Evidence

The following data is from an AWS credential report generated on January 1, 2023 showing the unused credentials:

Field	Value
user	exampleuser
arn	arn:aws:iam::111111111111:user/exampleuser
user_creation_time	2018-02-20T13:13:03+00:00
access_key_1_active	TRUE
access_key_1_last_rotated	2022-03-04T12:03:50+00:00
access_key_1_last_used_date	2022-03-20T08:16:00+00:00
access_key_1_last_used_region	us-east-1
access_key_1_last_used_service	ecs

Affected Resources

- arn:aws:iam::111111111111:user/exampleuser

Impact

Leaving unnecessary credentials enabled increases the window of opportunity for compromised credentials to be successfully used.

Recommendations

Audit unused credentials and remove them if they are no longer required.

Additional Information

- CWE-269: Improper Privilege Management

Cloud Network Penetration Test

6. MongoDB Service Does Not Require Authentication

■ SEVERITY: HIGH

Finding Information

A MongoDB instance was discovered that did not require authentication.

Evidence

The following command shows that the MongoDB service does not require authentication:

```
$ mongo 23.34.54.234
MongoDB shell version v3.6.8
connecting to: mongodb://23.34.54.234:27017/
Implicit session: session { "id" : UUID("77400b11-6742-4ead-9e9b-7a27e4057f4f") }
MongoDB server version: 4.0.5
WARNING: shell and server versions do not match
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2021-12-17T13:10:56.397+0000 I CONTROL  [initandlisten]
2021-12-17T13:10:56.397+0000 I CONTROL  [initandlisten] ** WARNING: Access
control is not enabled for the database.
2021-12-17T13:10:56.397+0000 I CONTROL  [initandlisten] **          Read and
write access to data and configuration is unrestricted.
2021-12-17T13:10:56.397+0000 I CONTROL  [initandlisten]
```

It was possible to list all the databases:

```
rs0:PRIMARY> show dbs

users
admin
config
local
```

Within the 'admin' database, an unauthenticated user can access the 'system.users' table and retrieve login names and hashes for all users with access to the database:

```
rs0:PRIMARY> db.system.users.find()

admin
tom
system
test
```

Affected Hosts

- 23.34.54.234:27017/tcp

Impact

An attacker on the network can access all information stored within the MongoDB service without entering a password. An attacker could also crack the hashes for a user's challenge question answers and use them to compromise or take over the account.

Recommendations

Enable authentication or restrict access to the MongoDB service.

Additional Information

- Authentication in MongoDB

7. Self-Signed Certificate

■ SEVERITY: MEDIUM

Finding Information

The server's TLS/SSL certificate is self-signed. Users have no way to verify the authenticity of the presented certificate. Using a self-signed certificate removes nearly all of the protections SSL is meant to provide.

Evidence

The following output from nmap shows that the certificate being used by the server is self-signed:

```
nmap 23.34.54.234 -p 443 --script +ssl-cert

Nmap scan report for 192.168.1.216
Host is up (0.00072s latency).

PORT     STATE SERVICE
443/tcp open  ms-wbt-server
| ssl-cert: Subject: commonName=dev-app.example.com
| Issuer: commonName=dev-app.example.com
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2022-10-02T06:38:45
| Not valid after:  2023-04-03T06:38:45
| MD5:   a18444b51063de212b3d333874adefa6
|_SHA-1: fb6e4e8e8db6e5ed75d68333043e8fd2220d0b98
MAC Address: 00:15:5D:01:3C:09 (Microsoft)
```

Affected Hosts

- 23.34.54.234:443/tcp

Impact

An attacker who is able to perform a man-in-the-middle attack can intercept and modify any communications between the client and the server by getting the client to accept the attacker's self-signed certificate instead of the server's self-signed certificate. This vulnerability is exploitable by any attacker who can perform a man-in-the-middle attack on the network traffic, which is generally any attacker on the local network.

Recommendations

Obtain a valid certificate that is signed by a trusted certificate authority.

Additional Information

- Wikipedia - Self-Signed Certificate

- The Dangers of Self-Signed SSL Certificates