



<Sample AWS>

Security Audit Report

Prepared by:

Senior Security Architect (architect@dataart.com)

Reviewed by:

Justin Surman (justin.surman@finstrides.com)

Limitations on Disclosure and Use of this Document

This document was prepared by DataArt for the exclusive benefit of Financial Strides and is proprietary information. Unauthorized use or reproduction of this document is prohibited. The Non-Disclosure agreement (NDA) in effect between DataArt and Financial Strides governs the disclosure of this document to all other parties, including product vendors or suppliers.

This report contains information about the configuration of <Sample AWS> platform and about its potential vulnerabilities. DataArt recommends that special precautions should be taken to protect the confidentiality of both document and the information contained herein. DataArt has retained and secured a copy of the document for customer reference. All other copies of the document have been delivered to Financial Strides.

By providing this report to Financial Strides, DataArt does not constitute any form of representation, warranty, or guarantee that the systems are 100% secure from every form of attack. While DataArt's methodology includes both automated and manual testing to identify and attempt exploitation of the most common security issues, testing was limited to an agreed upon timeframe.

Document Details

Title	Security Audit Report
Project Name	<Sample AWS>
Project Staff	<NAME SURNAME> – Senior Security Architect (OSCP OS-111-12345) <NAME SURNAME> – Offensive Security Lead (OSCP OS-111-56789) Justin Surman – Project Manager
Project Duration	01/05/2024 – 01/18/2024

Document History

Version	Date	Author	Comments
1.0	01/17/2024	<NAME SURNAME>	Initial version
1.1	01/18/2024	<NAME SURNAME>	Final review

Contents

Executive Summary	4
Assessment Methodology	7
Information Gathering	7
Terraform Security Code Review	7
AWS, Terraform and Gitlab Manual Assessment	8
Infrastructure Penetration Testing	9
Reporting	9
Criteria for Risk Ratings	10
Assessment Findings.....	11
Summary	11
High Risk Findings.....	12
H1. Lack of data protection at rest.....	12
H2. Lack of protection for CloudTrail log storage.....	14
Medium Risk Findings.....	16
M1. CloudWatch misconfiguration	16
M2. No container hardening procedures	17
M3. Vulnerable version of Gitlab	18
Low Risk Findings	22
L1. Password leakage	22
L2. Weak configuration of TLS	23
L3. Insecure communication between AWS components	26
L4. No access logging configured for S3 and ELB	27
Info Findings	29
IN1. Information disclosure in Gitlab pipelines	29
IN2. No Access Key rotation for IAM service accounts.....	30
IN3. Missing EBS and RDS backups	30
IN4. Misconfigured NACLs.....	32
IN5. Permissive rules for egress traffic in security groups	33
Conclusion	35

Executive Summary

Financial Strides engaged DataArt to perform security assessment of <Sample AWS> platform and associated infrastructure components and services. The primary goal of this exercise was to identify whether the configuration of AWS and related CI/CD tools had been set up in line with the industry security best practices and recommendations, as well as find out any misconfigurations and flaws which could lead to unauthorized access to any component of the setup.

The <Sample AWS> platform is a set of cloud components and environment templates which allows quick creation of cloud infrastructure from the scratch using Infrastructure as Code (IaC) approach. The platform uses Hub and Spoke topology with three spoke environments <ENV1, ENV2, ENV3> designed for containerized applications (ECS, ECR) and three more special AWS accounts created for DevOps tools (Gitlab), log aggregation (CloudTrail, CloudWatch) and security monitoring (AWS Config, AWS Trusted Advisor). All AWS accounts have built-in minimal security guardrails which are recommended by various security best practices; the platform could be augmented according to the needs of a particular client.

DataArt conducted security audit during the period of January 05 – 17, 2022. All assessment activities were performed on Pentest environment which was provisioned specifically for audit purposes. According to the plan agreed with the <COMPANY>, DataArt focused on the next activities:

- Security review of Terraform code responsible for creation of <Sample AWS>
- Assessment of secure provisioning of AWS services (Cloud Security Audit)
- Infrastructure pen test of the AWS pentest environment

During the infrastructure pen test, DataArt emulated an external attacker without prior knowledge of the environment (black-box testing), while the other activities were in scope of white-box testing. The testing did not attempt any active network-based DoS attacks. To conduct this project, <COMPANY>. supplied DataArt with the following assets:

- a set of AWS and VPN credentials to access the infrastructure;
- a separate VPN account to access Terraform source code.

The scope of the assessment included the following components, repositories, endpoints and tools:

- Terraform code stored in Gitlab EE
 - <https://gitlab.sampleapp.com/sample-aws-components>
 - <https://gitlab.sampleapp.com/sample-aws-blueprints>
 - <https://gitlab.sampleapp.com/sample-aws-demo>
 - <https://gitlab.sampleapp.com/sample-aws-security>
- AWS Pentest accounts accessible using AWS credentials
 - **sampleapp-demo-env1** (119115511411)

- [sampleapp-demo-env2](#) (533377677299)
- [sampleapp-demo-env3](#) (600600222777)
- [sampleapp-demo-logging](#) (360002288777)
- [sampleapp-demo-security](#) (895599595141)
- [sampleapp-demo-devops](#) (700922911454)
- Gitlab CE deployed in DevOps AWS account: <https://gitlab.demo.sampleapp-aws.net/>
- Public endpoints of the infrastructure
 - <https://git.sampleapp-aws.net/>
 - <https://demo-env1.demo.sampleapp-aws.net/>
 - <https://demo-env2.demo.sampleapp-aws.net/>
 - <https://demo-env3.demo.sampleapp-aws.net/>

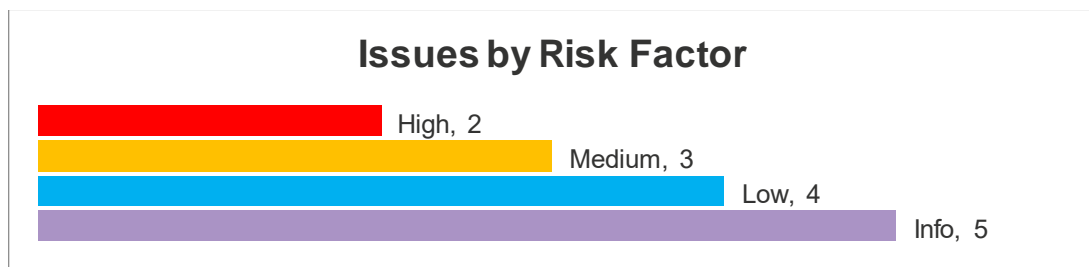
During the course of the assessment, DataArt was able to identify several potential security problems with the current configuration, including mismatches between actual configuration and recommended security defaults and potential security flaws.

Two major issues were caused by lack of controls for protecting confidentiality (encryption) and integrity (MFA Delete, Access Logging) of critical and sensitive data stored [by Gitlab](#) and [CloudTrail](#). Basic data protection procedures usually involve strong encryption, integrity controls and auditing; if all these mechanisms are missing or misconfigured, a malefactor could gain unauthorized access to confidential information and cover his/her traces by removing all audit trails.

In addition, there were three medium-severity issues caused by [absence of CloudWatch alarms for CloudTrail](#), usage of [Gitlab's version which had multiple vulnerabilities](#) listed in public databases of vulnerabilities (CVE, NVD) and [absence of hardening procedures for Docker images](#) which are produced in Gitlab pipelines and pushed into Elastic Container Registry (ECR).

DataArt has also identified a number of less critical issues and ways to improve the overall security of <Sample AWS> solution. A complete list of issues could be found in [Assessment Findings](#) section.

This report summarizes what DataArt believes are the most important issues to address in <Sample AWS> setup. The chart below outlines a number of issues identified that are grouped by risk factor. Note the risk ratings were given to help assist in prioritizing remediation efforts. True risk can only be calculated by an in-depth understanding of business processes and data, as well as the likelihood of exploitation.



DataArt recommends that <COMPANY>. addresses the findings contained in this report to minimize the attack surface available to an attacker and to ensure the overall security of the applications.

DataArt can re-verify remediated issues found during this assessment within 30 days of this report delivery.

Assessment Methodology

DataArt based the finding and recommendations, outlined in this report, on the information provided by the <Sample AWS> team members (either in form of interviews/demos or project artifacts), automated and manual Terraform code review, infrastructure penetration testing and manual and automated AWS configuration review.

Information Gathering

As the initial step of the assessment, DataArt collected the basic information about the environment and services and prepared an assessment checklist to track the status of the assessment and note any potential issues and deviations from the security baseline. DataArt used the next guidelines and recommendations for clouds to derive common rules and prepare the checklist:

- [Cloud Conformity Knowledge Base](#)
- [CIS Benchmark for Amazon Web Services](#)
- [Amazon IAM Best Practices](#)
- [Building a modular and Scalable Virtual Network Architecture with Amazon VPC, Best Practices](#)

The checklist included a number of items from the next major topics:

- Identity and Account Management (IAM)
- Audit, Logging and Monitoring
- Network Security
- Maintenance
- General Security

Terraform Security Code Review

DataArt used several open-source tools to perform automated code analysis and identify potential vulnerabilities and deviations from recommended defaults. The automated code analysis software inspects the code and identifies vulnerabilities, suspicious pieces of code and deviations from the adopted coding standard. The next tools were used to assess Terraform modules:

- **tfsec** (<https://github.com/liamg/tfsec>)
- **terrascan** (<https://github.com/cesar-rodriguez/terrascan>)

DataArt has also performed manual security code review of Terraform code and verified the results of automated tools to eliminate false positives. The scope of inspection included, but was not limited by the following security controls:

- Authentication mechanisms
- Credential and secret storage
- Transport layer security

- Data encryption at rest
- Network protection mechanisms
- Usage of secure protocols

Automated AWS Assessment

DataArt used several open-source tools to survey the targeted environment and identify potential vulnerabilities. To run these tools, DataArt obtained AWS CLI credentials for all AWS accounts. The next tools allowed to automate several complicated network checks, validate the existing configuration against the checklists and produce HTML reports with potential issues:

- Scout Suite (<https://github.com/nccgroup/ScoutSuite>)
- Prowler (<https://github.com/toniblyx/prowler>)

AWS, Terraform and Gitlab Manual Assessment

Using the information gathered during the previous stages, DataArt performed manual assessment of <Sample AWS> environment to identify any security and configuration issues and non-compliance with recommended security defaults. To be able to do that, DataArt used AWS accounts with *ReadOnly* AWS role assigned.

As part of manual assessment, DataArt has reviewed different AWS services, including, but not limited to, IAM, KMS, EC2, ECS, ECR, S3, VPC, Config, CloudWatch, CloudTrail, and performed the following actions:

- Observed types and placement of security controls
- Examined trust boundaries
- Reviewed authentication, session management and access controls
- Checked the separation of roles and duties
- Validated the data protection controls for securing data in transit and at rest
- Ensured the safety of remote and administrative access
- Identified attack detection and prevention mechanisms
- Verified the backup and data recovery policies and their implementations

As part of white box testing activities, DataArt has also analyzed Gitlab pipelines and Terraform code to identify any misconfigurations or weaknesses which might lead to information disclosure, account compromise or full control over the services.

As a result of manual assessment, DataArt has collected a number of evidences that the aforementioned security controls and policies are functioning as declared or have potential gaps and security flaws. These evidences include the screenshots of configurations, video recordings of interviews with the team members and other artifacts provided by the team.

Infrastructure Penetration Testing

During the white box infrastructure review, DataArt identified a number of public-facing endpoints and performed outside-in penetration testing of public services to find out any potential weaknesses which could be exploited by an attacker with sufficient motivation and level of skills. DataArt used several open source and commercial tools to survey the identified targets and identify potential vulnerabilities. The scope of testing included, but was not limited by the following:

- Parameter Injection
- Directory Traversal
- Parameter and Buffer Overflows
- Parameter Addition
- Path Manipulation
- Character Encoding
- Site Search
- SSL Strength
- Web Server/Web Package Identification
- Permissions Assessment
- Brute Force Authentication attacks

Using the information produced by the automated testing software, DataArt also employed manual testing techniques to identify and attempt exploiting additional vulnerabilities in the targeted infrastructure, and to eliminate false positives produced by the automated scanning process. The test focused on possible vulnerabilities in the application logic, looking for issues including but not limited to:

- Command injection
- Authentication and authorization implementation defects
- Privilege elevation
- Parameter overflow and handling
- HTTP/URL manipulation
- Improper web server configuration
- Information leakage
- SSL and transport layer weaknesses

Reporting

To summarize the findings and the risks associated with them, DataArt provided <COMPANY>. with this assessment report which describes the assessment findings in a formal, structured way and provides the recommendations regarding the remediation activities. The prioritized list of findings includes the next information for each item:

- Criticality
- Summary
- Affected Scope (environment, service, resource IDs, etc.)
- Evidences (including screenshots, configuration files, etc.)
- Recommendations on remediation

Criteria for Risk Ratings

The table below outlines the general rules for assigning risk ratings to identified vulnerabilities:

Risk Rating	Description
HIGH	<p>These issues identify faults in setups and environment misconfigurations that could significantly affect security and availability of networks, systems, applications or sensitive information, and directly result in compromising or gaining unauthorized access to networks, systems, applications or sensitive information.</p> <p>These issues could also describe significant gaps in processes and policies, like absence of firm-wide mandatory flows, processes and policies, which are not compensated by alternative flows, policies and controls.</p>
MEDIUM	<p>These issues identify faults in setups and environment misconfigurations that do not have an immediate or direct influence on security and availability of networks, systems, applications or information, but could provide a capability or information that, in combination with other capabilities or information, could affect security and availability of the whole system or result in compromising or gaining unauthorized access to a network, application or information.</p> <p>These issues could also describe moderate gaps in processes and policies, like absence of firm-wide recommended flows, processes and policies, which exist due to absence of controls which allow applying and executing them.</p>
LOW	<p>These issues identify faults in setups and environment misconfigurations that do not affect security and availability of networks, systems, applications or information, but could provide information that might be used in combination with other information to gain insight into how to compromise or gain unauthorized access to networks, systems, applications or information or to leave information security and incident response teams without extra information about incidents, like logs, audit trails, etc.</p> <p>These issues could also describe minor and moderate gaps in processes and policies, like misalignments with recommended best practices, firm-wide policies and recommendations, which may exist due to absence of controls which allow applying and executing them, and which are partially compensated by alternative mechanisms.</p>

INFO

These findings should not be considered as vulnerabilities or issues right now; however, they should be taken into account to improve overall security of a setup or environment in the future.

Assessment Findings

Summary

The table below outlines summary of findings identified during penetration testing:

Finding Description	Status
High Risk Findings	
H1. Lack of data protection at rest	HIGH
H2. Lack of protection for CloudTrail log storage	HIGH
Medium Risk Findings	
M1. CloudWatch misconfiguration	MEDIUM
M2. No container hardening procedures	MEDIUM
M3. Vulnerable version of Gitlab	MEDIUM
Low Risk Findings	
L1. Password leakage	LOW
L2. Weak configuration of TLS	LOW
L3. Insecure communication between AWS components	LOW
L4. No access logging configured for S3 and ELB	LOW
Info Findings	
IN1. Information disclosure in Gitlab pipelines	INFO
IN2. No Access Key rotation for IAM service accounts	INFO

IN3. Missing EBS and RDS backups	INFO
IN4. Misconfigured NACLs	INFO
IN5. Permissive rules for egress traffic in security groups	INFO

High Risk Findings

Two **high**-severity issues were found in <Sample AWS> configuration, as described below.

H1. Lack of data protection at rest

Risk Rating: **HIGH**

Summary

Data protection at rest aims to secure inactive data stored on any device or network and helps companies or other controlling parties ensure that stored data is not vulnerable to hacking or other unauthorized access. Basic data at rest protection procedures usually involve strong data encryption.

While evaluating encryption mechanisms used in <Sample app> Pentest environment, DataArt noticed that some critical components, like S3 buckets containing sensitive log information and EBS volume of Gitlab instance, were not using either built-in or custom data encryption mechanism. Absence of data encryption at rest increases the risk of unauthorized access to confidential information and its accidental disclosure.

Affected Scope

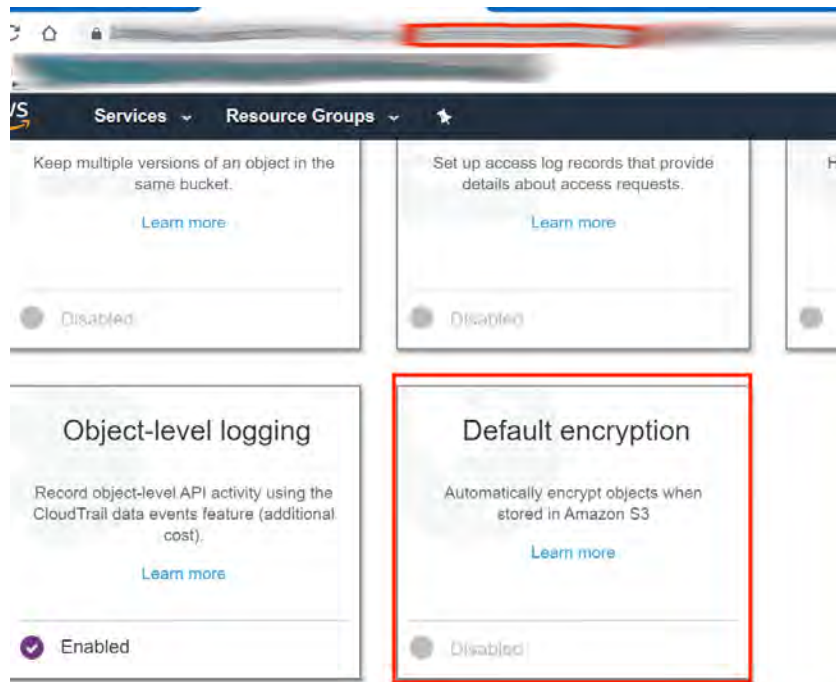
S3 buckets in **sampleapp-demo-logging** AWS account:

- **sampleapp-demo-cfg-logs-<random suffix>**
- **sampleapp-demo-gdd-logs-<random suffix>**

Terraform module **tf-aws-gitlab-ce/application_server.tf**

Evidences

A screenshot of **sampleapp-demo-cfg-logs** S3 bucket's configuration page. **Default Encryption** was disabled:



The same issue was discovered during Terraform code review. Here's a piece of Terraform code responsible for creation of an EBS volume for Gitlab instance. Encryption was not enforced here:

```
resource "aws_launch_configuration" "application_server" {
  name_prefix         = local.application_server
  instance_type       = var.application_server.instance_type
  image_id            = var.application_server.ami_id
  key_name            = aws_key_pair.gitlab.key_name
  associate_public_ip_address = false
  iam_instance_profile = aws_iam_instance_profile.application_server_iam_profile.name

  root_block_device {
    volume_size = 30
    volume_type = "gp2"
    delete_on_termination = true
  }

  security_groups = [
    aws_security_group.application_server.id,
    var.postgres.user_security_group,
    var.elasticache.client_security_group_id,
  ]
}
```

Finally, here's one more Terraform template related to S3 buckets. ***Server_side_encryption_configuration*** block is missing here as well:

```
resource "aws_s3_bucket" "bucket" {  
  bucket = "${var.bucket_name}${var.append_random_suffix ? "-${random_string.random_  
suffix.result}" : ""}"  
  
  force_destroy = var.force_s3_destroy  
  
  tags = {  
      
  }  
}  
  
resource "random_string" "random_suffix" {  
  length      = 12  
  upper       = false  
  lower       = true  
  number      = true  
  special     = false  
}
```

Recommendations

DataArt recommends following security best practices described in Cloud Conformity KB – it's necessary to set up [S3 bucket default encryption](#) and set up [customer-managed keys \(CMKs\)](#) for being used by AWS KMS to secure data in S3. It's also possible to use [S3 Server-side encryption \(SSE\)](#) to perform transparent encryption and decryption while the data is stored and fetched, respectively. EBS volumes holding critical data, like source code, [should be encrypted](#) as well.

H2. Lack of protection for CloudTrail log storage

Risk Rating: **HIGH**

Summary

AWS CloudTrail is a service that helps to monitor, survey, and perform operation auditing along with risk monitoring of AWS accounts. With AWS CloudTrail, the user will be able to log, ceaselessly monitor, and retain account activity associated with actions across the AWS infrastructure – in other words, CloudTrail provides the complete account activity of the Amazon Web Services. Such event history simplifies security analysis, resource amendment trailing, and troubleshooting.

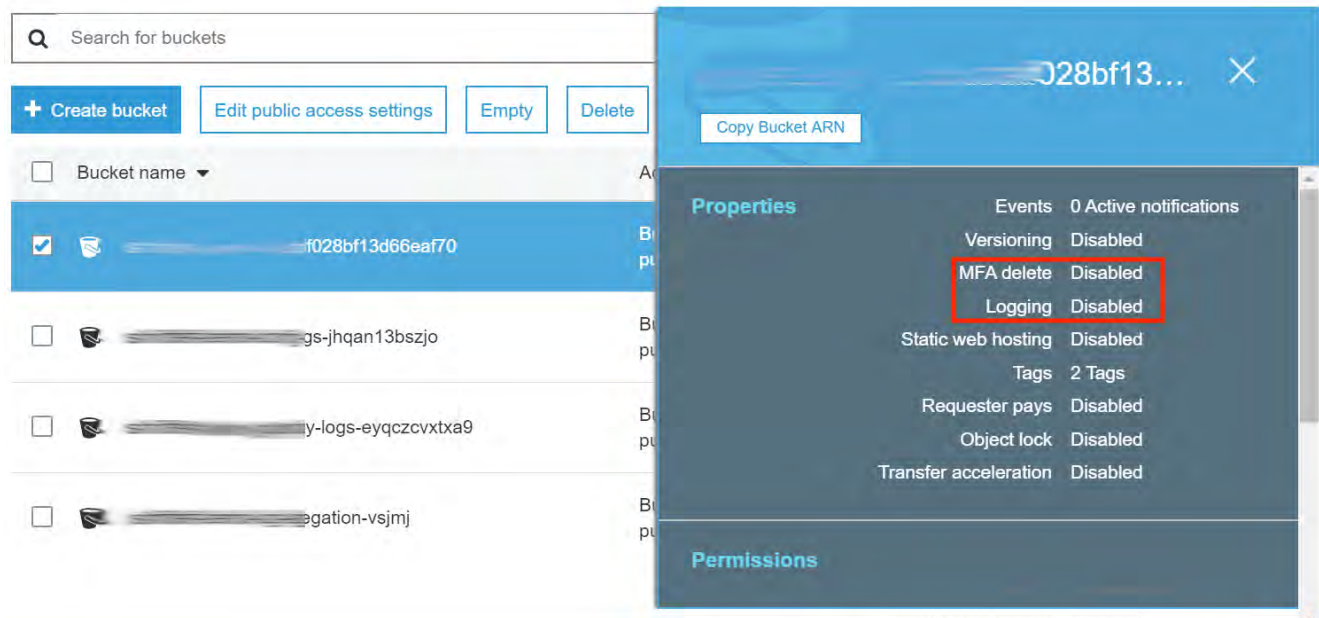
A common setup is to have multiple trails from several AWS accounts delivering log files to one Amazon S3 bucket, and such logging strategy is implemented in <Sample AWS>. While checking CloudTrail security controls, DataArt found out that some security controls, like MFA Delete and Server Access Logging for buckets containing CloudTrail logs, were not enabled, thus leaving a way for a malefactor, who has gained administrative access to the environment, to cover his/her traces.

Affected Scope

sampleapp-demo-logging AWS account, S3 bucket **sampleapp-demo-cloudtrail<random suffix>**

Evidences

The next image shows the configuration of S3 bucket which was used as storage for CloudTrail logs. Both MFA Delete and Server Access Logging were disabled:



Recommendations

According to Cloud Conformity knowledge base, it's necessary to [enable MFA Delete](#) for S3 buckets containing CloudTrail logs, so it would be impossible to delete the buckets without providing valid MFA codes. Apart from that, it's recommended to switch [Server Access Logging](#) on for these buckets in order to track requests for accessing the buckets and collect them for further usage in security audits and incident response workflows.

Medium Risk Findings

Three **medium**-severity issues were found in <Sample AWS> configuration, as described below.

M1. CloudWatch misconfiguration

Risk Rating: **MEDIUM**

Summary

According to security best practices, it's necessary to monitor important security events, like authentication and authorization successes and failures, critical configuration changes, use of high-risk functionality, etc. In absence of external security information and event management (SIEM) systems, AWS CloudTrail events could be monitored with CloudWatch Logs for management and security purposes.

While checking audit and alerting controls, DataArt revealed that integration between CloudTrail and CloudWatch was not set up, so, as a result, there were no CloudWatch alarms configured for CloudTrail logs. Absence of alerting for critical operational events detected via CloudTrail log analysis increases the response time to these events.

Affected Scope

sampleapp-demo-logging AWS account, CloudTrail and CloudWatch





Evidences

Here's a picture of CloudTrail configuration used in **sampleapp-demo-logging** AWS account of <Sample AWS> Pentest environment. CloudWatch Logs feature had not been set up, so Log Group parameter was empty:

Trails

Deliver logs to an Amazon S3 bucket. CloudTrail events can be processed by one trail for free. There is a charge for processing events with additional trails. For more information, see [AWS CloudTrail Pricing](#).

Create trail

Trail name	Home region	Multi-region trail	Insights	Organization trail	S3 bucket	Log file prefix	CloudWatch Logs Log group	Status
	US East (N. Virginia)	Yes	Disabled	No	 70	logging		

Recommendations

According to security best practices described in CIS Benchmark for Amazon and Cloud Conformity KB, it's recommended to configure [integration between CloudTrail and CloudWatch Logs](#) and set up the [following security alerts](#):

- Usage of root user

- AWS Management Console sign-in without MFA
- AWS Management Console authentication failures
- Unauthorized API calls
- IAM policy changes
- CloudTrail configuration changes
- S3 bucket policy changes
- AWS Config configuration changes
- Security group changes
- VPC, NACL, gateway and route table changes
- Disabling or scheduled deletion of customer created CMKs

CloudWatch-based alerting configuration will be popular among those clients which do not have any external SIEM systems and will also satisfy basic security and compliance requirements.

M2. No container hardening procedures

Risk Rating: **MEDIUM**

Summary

During the Q&A session with <Sample AWS> team members, DataArt asked a question about the hardening procedures used for containers produced by Gitlab EE pipelines before they're uploaded into ECR and whether there are any benchmarks executed to verify how security best practices are applied to containers. The interviewers noted that the team is not using any security verification and environment hardening procedures for Docker containers at the moment. Absence of security hardening procedures for containers and container orchestration systems is not compliant with Security in Depth principle which is essential for mission-critical production environments.

Affected Scope

All ECR containers in the following AWS accounts:

- **sampleapp-demo-env1**
- **sampleapp-demo-en2**
- **sampleapp-demo-env3**

Evidences

Q&A Interview session recording (25:10) – <Interviewee 1> and <Interviewee 2> mentioned about absence of environment hardening procedures.

Recommendations

DataArt recommends using [CIS Docker Benchmark](#) for assessing the security of containers produced in CI/CD pipelines and performing basic environment hardening depending on the capabilities used by containers. This security verification measure could be automated by executing all checks with help of a [Docker container](#) which is

deployed in the same cluster/subnet with the containers being assessed. The benchmark could also help in setting up additional security guardrails according to industry best practices.

M3. Vulnerable version of Gitlab

Risk Rating: **MEDIUM**

Summary

Known vulnerabilities are vulnerabilities that were discovered in third party components and published in the NVD, security advisories or issue trackers. Such vulnerabilities could be exploited by hackers who find the relevant documentation and are able to construct various exploitation mechanisms for the issues. The problem of using components with known vulnerabilities is highly prevalent ([it is reflected in OWASP's Top Ten 2017, A9](#)), and the possible impact of issues in 3rd party dependencies could range from minor to some of the largest breaches known.

DataArt observed that Gitlab CE deployed at <https://gitlab.demo.sampleapp-aws.net/> used an outdated version (11.0.2) which is known to have three vulnerabilities listed in CVE. Two of the issues listed below are exploitable:

- [CVE-2019-5467](#) – Exploitable
- [CVE-2019-11605](#)
- [CVE-2019-15724](#) – Exploitable

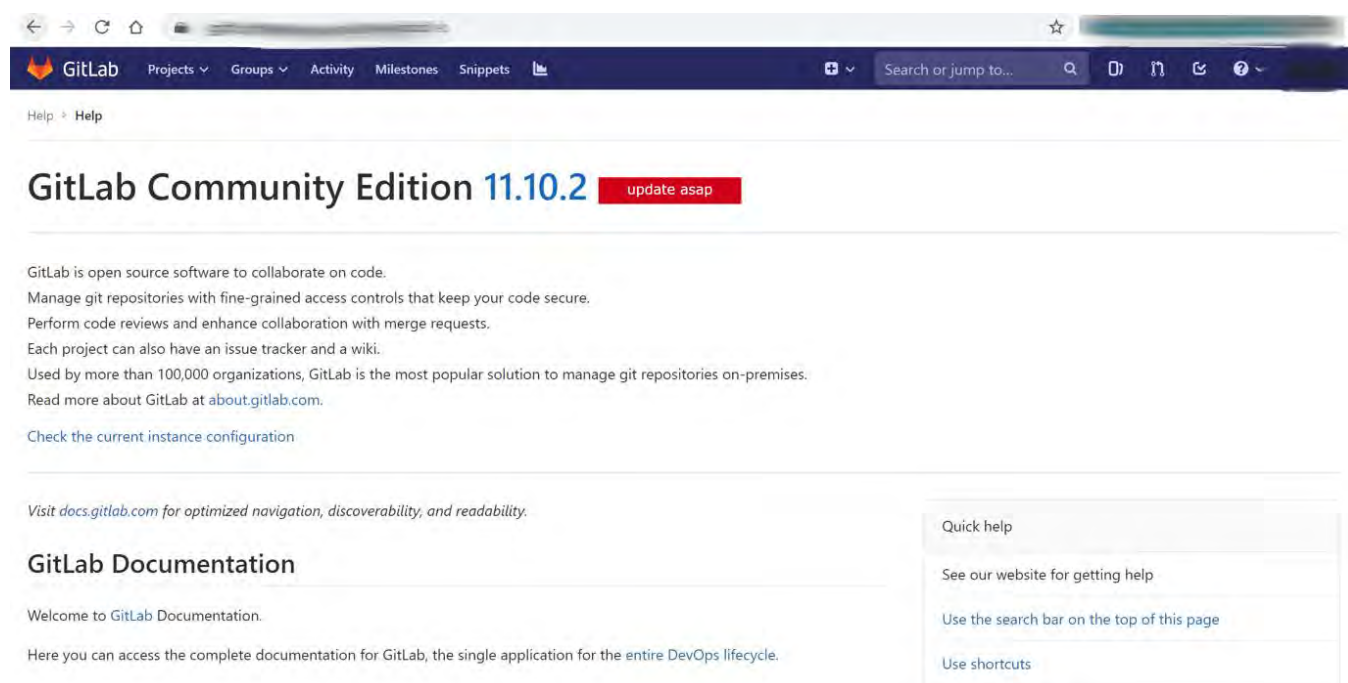
Please note that the severity of this finding was set up according to the severities of related CVEs.

Affected Scope

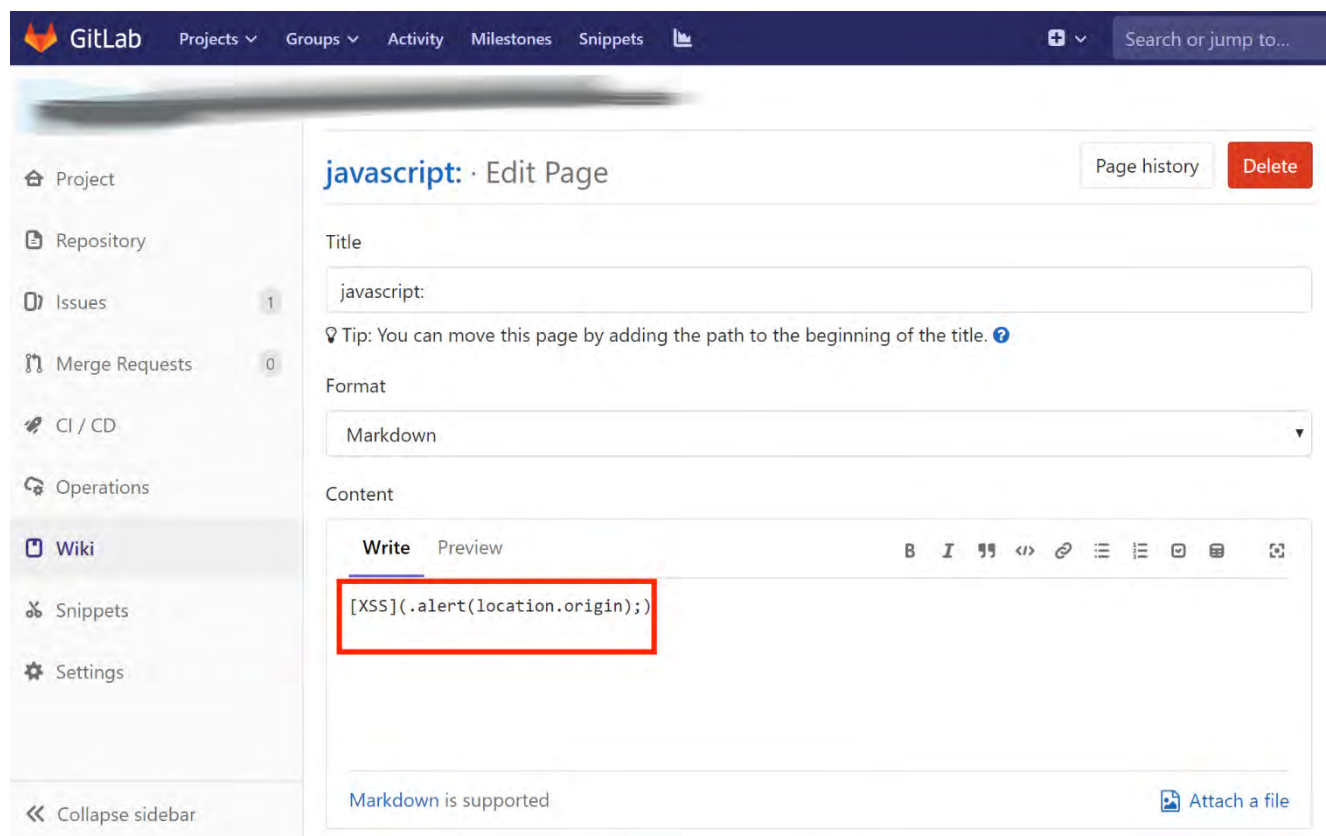
Gitlab at <https://gitlab.demo.sampleapp-aws.net/>

Evidences

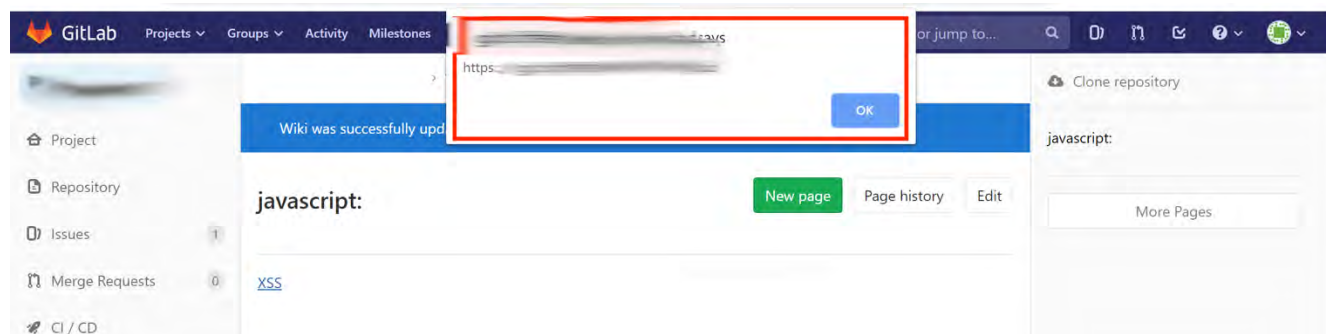
An image of Gitlab's Help page – the software itself asks for an immediate update and warns the users with "UPDATE ASAP" red status plate:



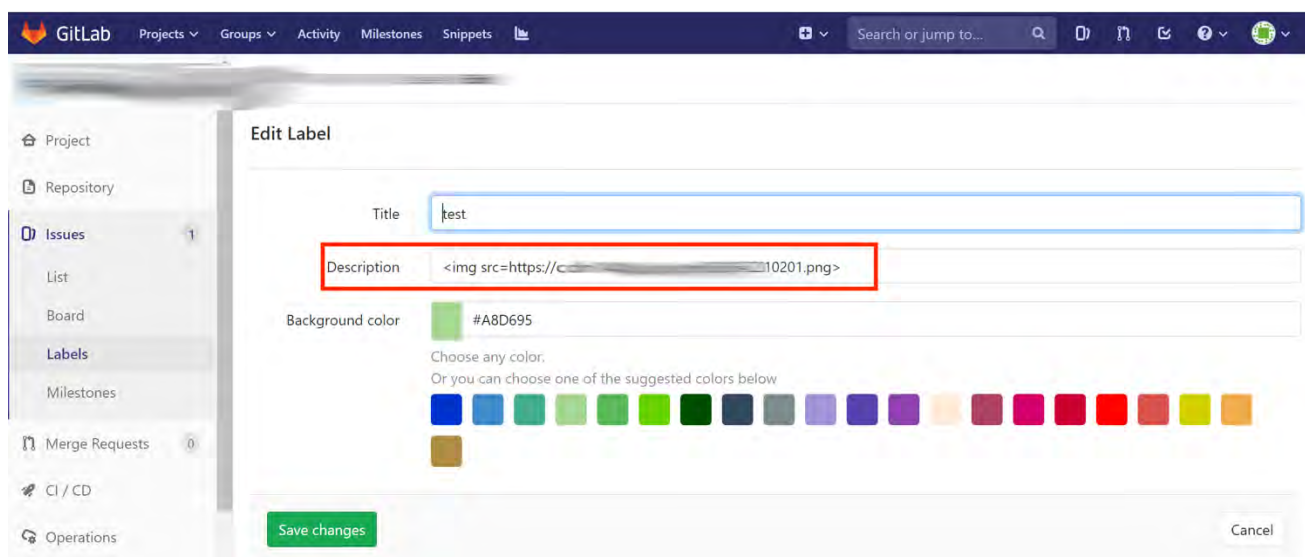
Here are the screenshots which demonstrate how a stored XSS vulnerability ([CVE-2019-5467](https://cve.org/CVERecord?id=CVE-2019-5467)) could be exploited (full step-by-step description of exploitation process could be found at [this resource](#)). The first image describes how to create a stored XSS:



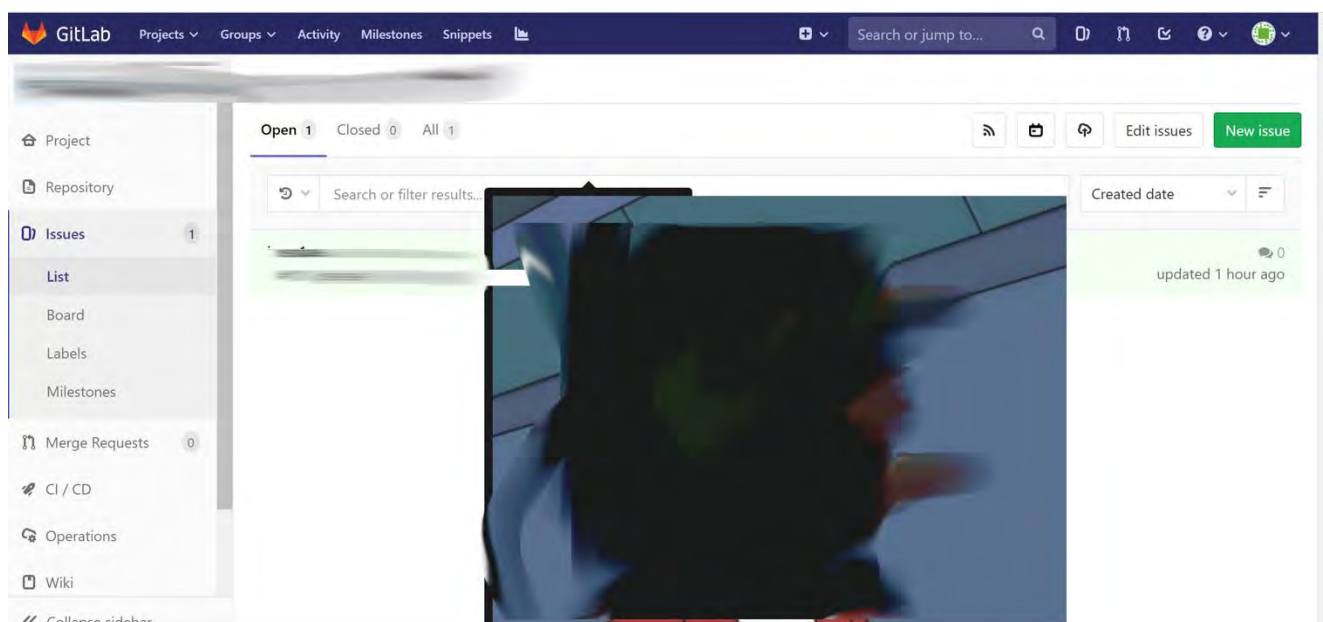
The second one shows the result of the injection:



And the following images prove how HTML Injection vulnerability of Gitlab's labels ([CVE-2019-15724](#)) could be used. A complete description of how the issue could be reproduced is available [in this ticket](#). This picture shows the process of injection of an external URL:



The next one illustrates the result of the injection:



Recommendations

According to security best practices, it's necessary to keep software components up to date in order to receive latest security updates and patches, so DataArt recommends updating Gitlab to the most recent version as soon as possible.

Low Risk Findings

Four **low**-severity issues were found in <Sample AWS> configuration, as described below.

L1. Password leakage

Risk Rating: **LOW**

Summary

Passwords, API keys, authentication tokens and other types of credentials must be stored and used securely to prevent unintended leaks and further account hijacks. Usage of passwords in plaintext in non-interactive pipelines, as well as storage of credentials in plaintext in configuration files and environment variables imposes huge security risks, since the secrets are not protected by any means and could be easily stolen by a malefactor who gained access to configuration files or shell history.

While checking Gitlab pipelines, DataArt found out that *push_image* step invoked *aws ecr* command-line tool which resulted in a response which contained a Docker login command with insecure *--password* option, and the same password was stored in Docker's config file in plaintext. If a malefactor gains access to Gitlab runner machine, he/she will be able to use cached credentials and access AWS ECR registry on behalf of Gitlab and push malicious Docker images instead of valid ones.

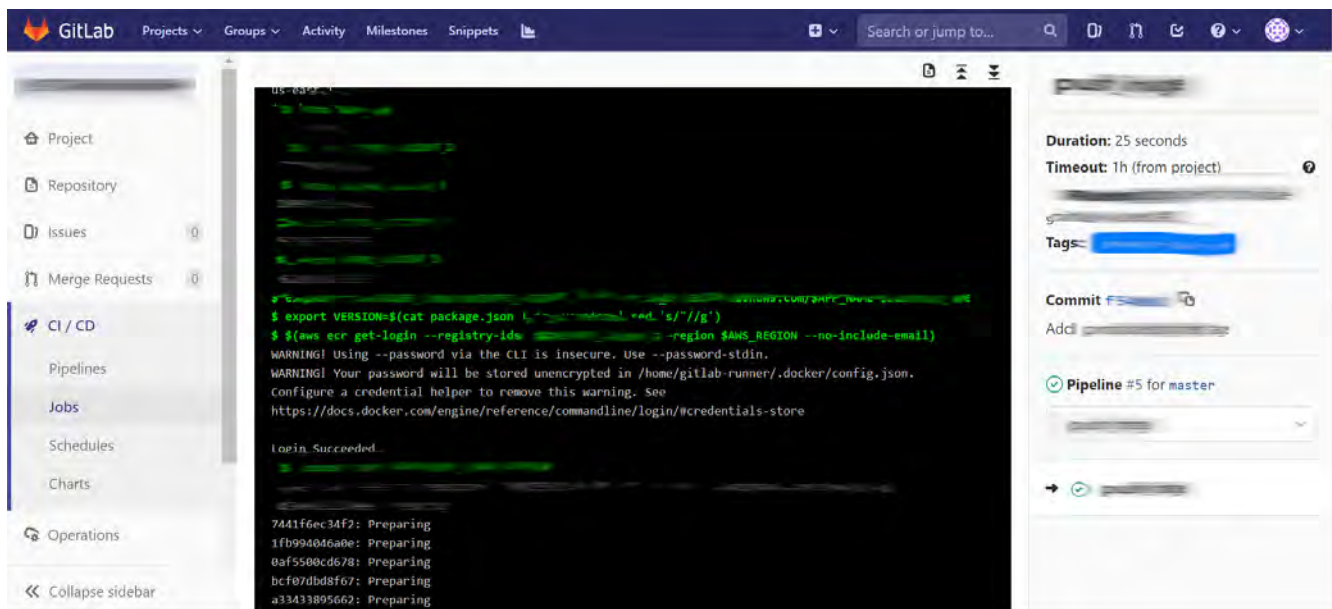
Since DataArt was not able to gain access to cached credentials, the original risk rating was reduced to Low.

Affected Scope

push_image step of Gitlab EE CI/CD pipelines

Evidences

Here's a screenshot of Gitlab's *push_image* pipeline step with full output of the step's script. There are two warnings from Docker engine highlighted:



Recommendations

[According to official Docker's documentation](#), it's recommended to use `--password-stdin` option for non-interactive login into container registry and configure a system-native credential storage to secure credentials properly. [A known issue](#) with lack of support for `--password-stdin` option in AWS ECR CLI could be bypassed by using [this workaround](#).

L2. Weak configuration of TLS

Risk Rating: **LOW**

Summary

Transport Layer Security is commonly used in websites and web applications together with the HTTP protocol to secure the transfer of data by applying encryption and digital signatures. TLS uses cipher suites, which are combinations of authentication, encryption, and message authentication code (MAC) algorithms, to negotiate security settings for a TLS connection as well as for the transfer of data. Usage of outdated versions of TLS and bad TLS configurations could make websites and web applications vulnerable to various attacks, like POODLE and BEAST.

During the infrastructure penetration test and cloud audit, DataArt noticed that all Application Load Balancers (ALBs) at AWS were using old security policies (namely ELBSecurityPolicy-2016-08) which allowed connection downgrade to TLS 1.0 and TLS 1.1. At the same time, all ALBs and Gitlab were also using a number of TLS 1.2 cipher suites which are considered weak nowadays due to known security concerns associated with the cryptographic algorithms used in those suites:

- RSA key agreement protocol does not support Perfect forward secrecy (FS, or PFS) – feature of specific key agreement protocols that gives assurances that session keys will not be compromised even if the private key of the server is compromised.

- AES encryption in Cipher Block Chaining (CBC) mode is vulnerable to padding oracle attacks – variable-length plaintext messages have to be padded (expanded) to be compatible with the underlying cryptographic primitive, so the attack relies on having an "oracle" who freely responds to queries about whether a message is correctly padded or not.
- HMAC-SHA1 is unsafe since SHA-1 is vulnerable to collision attacks – a group of researchers [has recently broken SHA-1 in practice](#).


Affected Scope

All public endpoints of <Sample AWS> environment:

- <https://git.sampleapp-aws.net/>
- <https://demo-env1.demo.sampleapp-aws.net/>
- https://demo-env2.demo.sampleapp-aws.net
- https://demo-env3.demo.sampleapp-aws.net

Evidences

Here's a screenshot from SSL Labs (<https://www.ssllabs.com>) scanning results for Gitlab endpoint. Almost all cipher suites (except those ones based on AES-GCM) are marked as weak:



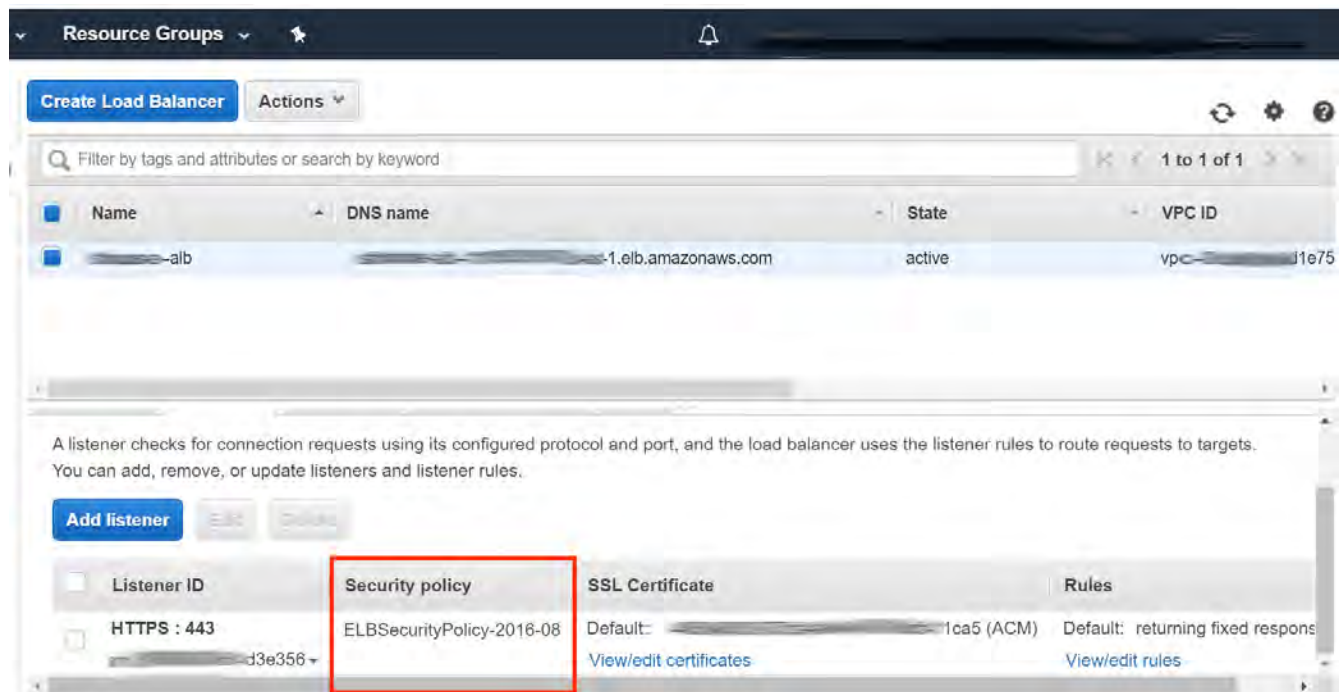
Cipher Suites			
# TLS 1.2 (suites in server-preferred order)			
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)	ECDH secp256r1 (eq. 3072 bits RSA) FS		128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)	ECDH secp256r1 (eq. 3072 bits RSA) FS	WEAK	128
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)	ECDH secp256r1 (eq. 3072 bits RSA) FS		256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)	ECDH secp256r1 (eq. 3072 bits RSA) FS	WEAK	256
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)		WEAK	128
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)		WEAK	128
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)		WEAK	256
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)		WEAK	256

The next picture represents scanning results produced by *ssllscan* utility for ALB in Env1 environment (both Env2 and Env3 had identical SSL settings). Both TLS 1.0 and TLS 1.1 are not disabled, and TLS 1.2 supports multiple weak cipher suites as well. There were only two safe cipher suites (highlighted in green):


```
Windows PowerShell
PS C:\Users\user> .\targets.ssllscan.txt

OpenSSL 1.0.2=rhacha (1.0.2g-dev)
connected to 10.0.0.1:443
Testing SSL server
  TLS Fallback scsv:
Server supports TLS Fallback scsv
  TLS renegotiation:
Secure session renegotiation supported
  TLS Compression:
compression disabled
  Heartbleed:
TLS 1.2 not vulnerable to heartbleed
TLS 1.1 not vulnerable to heartbleed
TLS 1.0 not vulnerable to heartbleed
Supported Server Cipher(s):
Preferred TLSv1.2 128 bits ECDHE-RSA-AES128-GCM-SHA256 Curve P-256 DHE 256
Accepted TLSv1.2 128 bits ECDHE-RSA-AES128-SHA256 Curve P-256 DHE 256
Accepted TLSv1.2 128 bits ECDHE-RSA-AES128-SHA Curve P-256 DHE 256
Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-GCM-SHA384 Curve P-256 DHE 256
Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-SHA384 Curve P-256 DHE 256
Accepted TLSv1.2 256 bits ECDHE-RSA-AES256-SHA Curve P-256 DHE 256
Accepted TLSv1.2 128 bits AES128-GCM-SHA256
Accepted TLSv1.2 128 bits AES128-SHA
Accepted TLSv1.2 256 bits AES256-GCM-SHA384
Accepted TLSv1.2 256 bits AES256-SHA256
Accepted TLSv1.2 256 bits AES256-SHA
Preferred TLSv1.1 128 bits ECDHE-RSA-AES128-SHA Curve P-256 DHE 256
Accepted TLSv1.1 256 bits ECDHE-RSA-AES256-SHA Curve P-256 DHE 256
Accepted TLSv1.1 128 bits AES128-SHA
Accepted TLSv1.1 256 bits AES256-SHA
Preferred TLSv1.0 128 bits ECDHE-RSA-AES128-SHA Curve P-256 DHE 256
Accepted TLSv1.0 256 bits ECDHE-RSA-AES256-SHA Curve P-256 DHE 256
```

Such TLS configuration is supported in the next ELB Security policy (see the screenshot for Env1 environment below, Env2 and Env3 had identical SSL settings), which should be considered unsafe:



Recommendations

DataArt suggests using stricter TLS policies for application load balancers (like [ELBSecurityPolicy-FS-1-2-Res-2019-08](#)), which do not allow fallback to TLS 1.0 and TLS 1.1, and disabling all cipher suites which fall under one of the following conditions:

- RSA is used for key agreement
- AES-CBC is used for data encryption
- HMAC-SHA1 (SHA) is used for message authentication

L3. Insecure communication between AWS components

Risk Rating: **LOW**

Summary

Data protection in transit helps keeping data confidentiality and integrity while it's traveling from network to network or being transferred from a local storage device to a cloud storage device. Effective data protection measures for in transit data are critical as data is often considered less secure while in motion. Organizations that fail to protect data in transit are more susceptible to man-in-the-middle attacks, eavesdropping, and session hijacking.

DataArt revealed that the communication channels between application load balancers (ALBs) and containers deployed at Fargate (ECS) were not protected by means of encryption. At the same time, connections from containers to data storages in RDS and S3 were using secure protocols, so ALB—Fargate communication became the weakest link in the full data transfer chain (from end user to storage and vice versa). A potential malefactor with network-level access to VPC could sniff the traffic or perform active attacks.

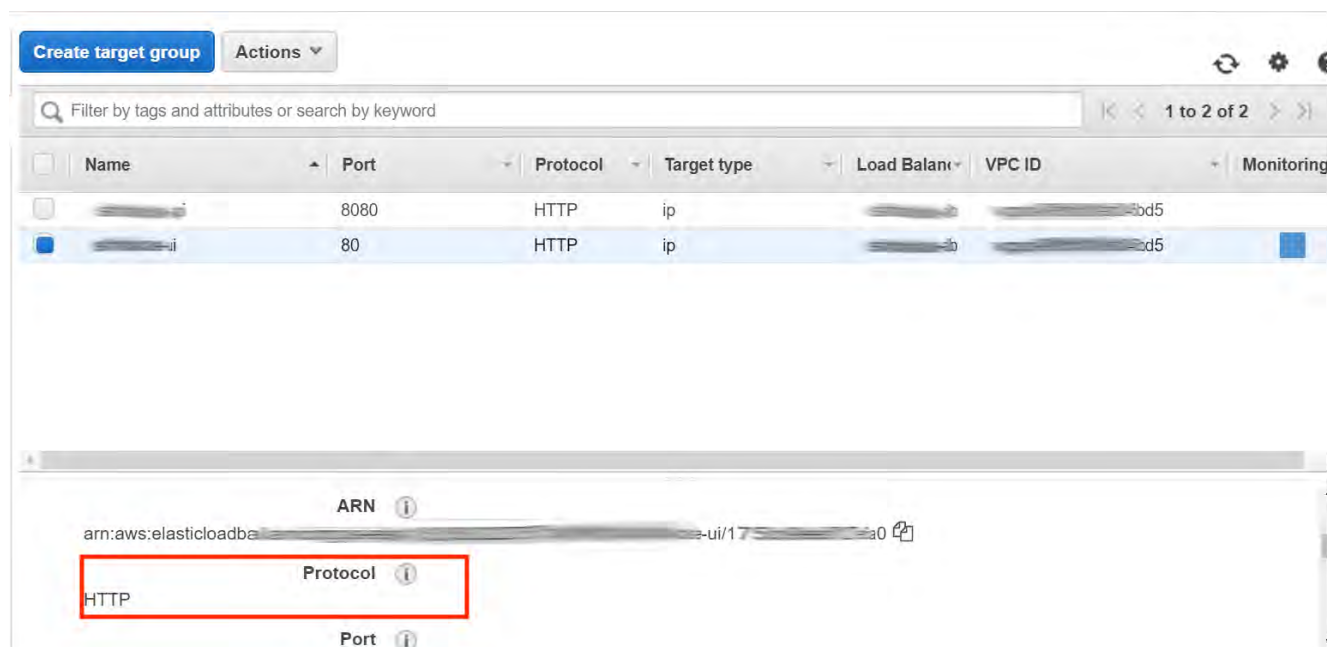
Affected Scope

All ALB Target groups in the following AWS accounts:

- **sampleapp-demo-env1**
- **sampleapp-demo-env2**
- **sampleapp-demo-env3**

Evidences

Here's a screenshot which shows ALB's target groups in Env3 environment. Both **app1-api** and **app1-ui** used cleartext HTTP without encryption:



Recommendations

According to Security in Depth principle, it's necessary to configure encrypted communication channel between load balancers and Fargate clusters to eliminate the weakest link in data transfer chain.

L4. No access logging configured for S3 and ELB

Risk Rating: **LOW**

Summary

Access logs capture detailed information about requests sent to various pieces of infrastructure. Each access log entry contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses, so these logs could be used to analyze various patterns and troubleshoot issues.

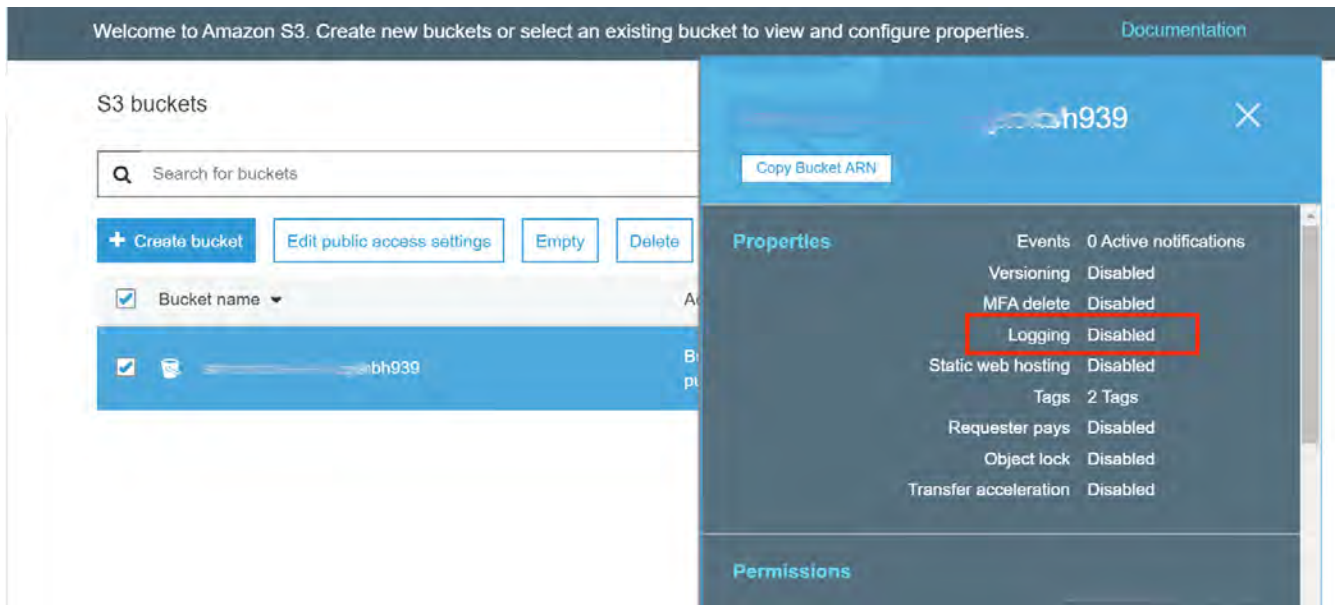
DataArt noted that all S3 buckets and Application Load Balancers had access logging feature disabled. Absence of access logging increases the risk of skipping important security events or unauthorized access to storage buckets and may complicate the investigation of security incidents and performance pitfalls.

Affected Scope

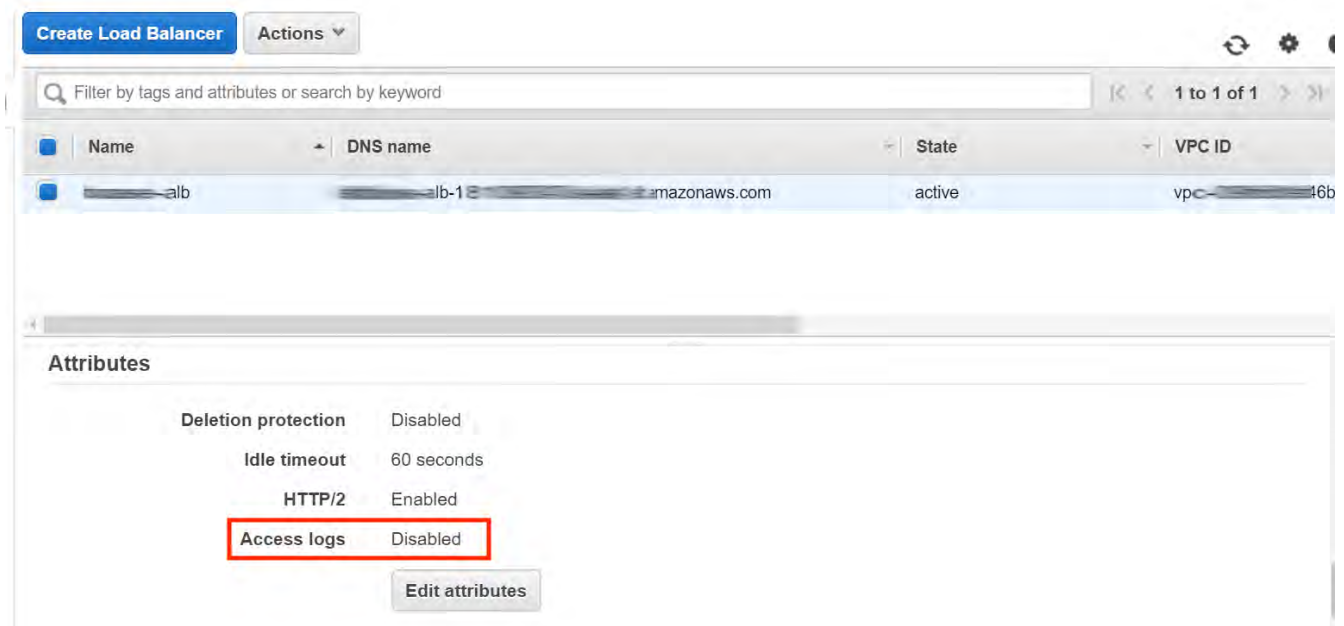
All ALBs and S3 buckets in all <Sample AWS> accounts

Evidences

The picture below confirms the fact of absence of S3 access logging for *aaaa-test-<random id>* bucket in *sampleapp-demo-env3* AWS account:



Another image shows that *showcase-alb* in *sampleapp-demo-env3* AWS account lacks access logging as well:



Recommendations

DataArt recommends following security best practices listed in CIS Benchmark and Cloud Conformity KB and suggests switching both S3 Server Access Logging and ALB Logging on in order to record access requests which might be useful for security audits and incident response workflows and perform log analysis to define and implement extra security controls against unauthorized access to data.

Info Findings

Five **info** findings were identified in <Sample AWS> configuration, as described below.

IN1. Information disclosure in Gitlab pipelines

Risk Rating: **INFO**

Summary

Information disclosure usually happens when an application or a service fails to properly protect sensitive and confidential information from parties that are usually not supposed to have access to the subject matter. Though such issues are not exploitable in most cases, they are considered as real threat because they allow malefactors to gather relevant information which can be used later in an attack.

While assessing Gitlab CI/CD pipelines, DataArt observed that all deployment steps launched a script which dumped JSON objects describing the infrastructure deployed in AWS, thus leaving this potentially sensitive information disclosed for people who are probably not authorized to see it.

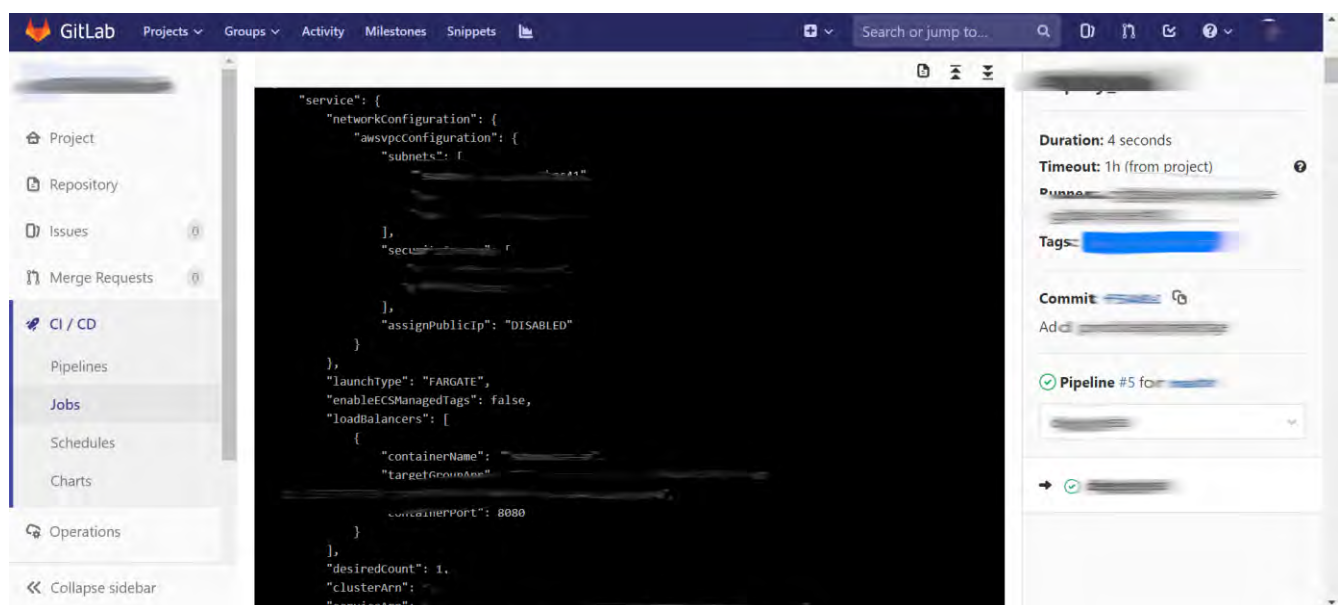
Affected Scope

All deployment steps of Gitlab CE pipelines:

- `test_deploy_env1`
- `test_deploy_env2`
- `test_deploy_env3`

Evidences

Here's a screenshot of `test_deploy_env2` step of Gitlab's CI/CD pipeline. The output of the command contained the up-to-date description of different AWS services, including EC2, VPC, ECS and others:



Recommendations

Since information about cloud infrastructure could be used by malefactors to prepare targeted attacks, it's necessary to reduce the verbosity of deployment steps of Gitlab pipelines in order to avoid further disclosures of potentially sensitive data.

IN2. No Access Key rotation for IAM service accounts

Risk Rating: INFO

Summary

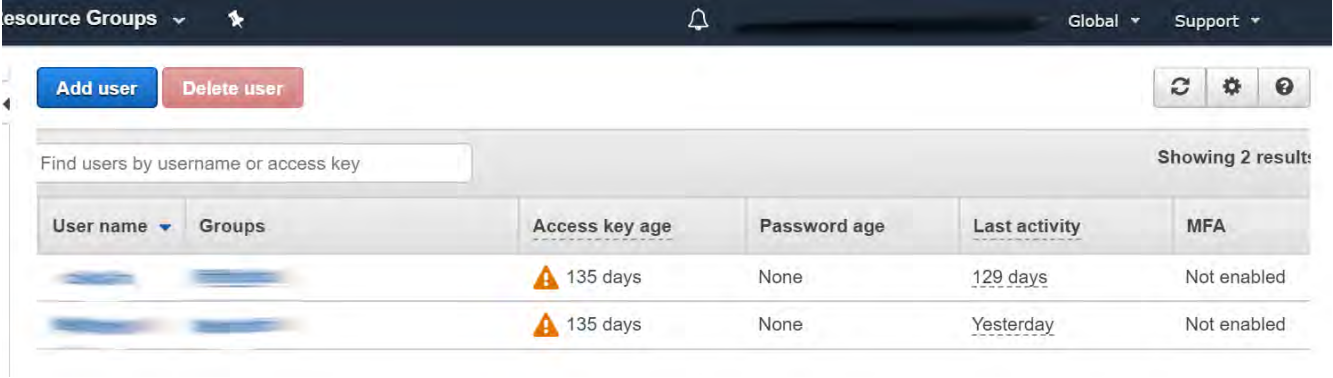
Periodical rotation of Identity and Access Management (IAM) credentials significantly reduces the chances that a compromised set of access keys can be used to access certain components within AWS accounts. While auditing IAM console, DataArt identified that access keys for two service accounts in each AWS account were rotated more than 90 days ago. Usage of outdated access keys increases the likelihood of stolen credentials attacks and unauthorized access to AWS resources.

Affected Scope





All <Sampe AWS> service accounts (*sso-temp* and *terraform-cli*)

Evidences

A screenshot which shows the ages of access keys for *sso-temp* and *terraform-cli* accounts in Env1 AWS account:



The screenshot shows the AWS IAM console interface. At the top, there are tabs for 'Resource Groups' and 'Users'. Below the tabs, there are buttons for 'Add user' and 'Delete user'. A search bar is present with the text 'Find users by username or access key'. To the right of the search bar, it says 'Showing 2 results'. Below the search bar, there is a table with the following columns: 'User name', 'Groups', 'Access key age', 'Password age', 'Last activity', and 'MFA'. The table contains two rows of data. The first row shows a user with an access key age of 135 days, no password age, last activity 129 days ago, and MFA not enabled. The second row shows a user with an access key age of 135 days, no password age, last activity yesterday, and MFA not enabled.

User name	Groups	Access key age	Password age	Last activity	MFA
		⚠️ 135 days	None	129 days	Not enabled
		⚠️ 135 days	None	Yesterday	Not enabled

Recommendations

Even though *sso-temp* will never appear in client's setups and *terraform* account will probably be deleted after environment provisioning, it's highly recommended to use an automated workflow for disabling access keys automatically after X days (recommended value for X is 90 days). DataArt suggests using a lambda function named [AWS Key Disabler](#) for this purpose.

IN3. Missing EBS and RDS backups

Risk Rating: INFO

Summary

Keeping reliable backups is an important part of security as high availability. Regular backups protect against the risk of damage or loss due to hardware failure, software or media faults, viruses or hacking, power failure and human errors.

DataArt found out that there were no RDS automated backups and EBS snapshots. Absence of recent backups could result in impossibility to recover the whole environment and significant data loss.

Affected Scope

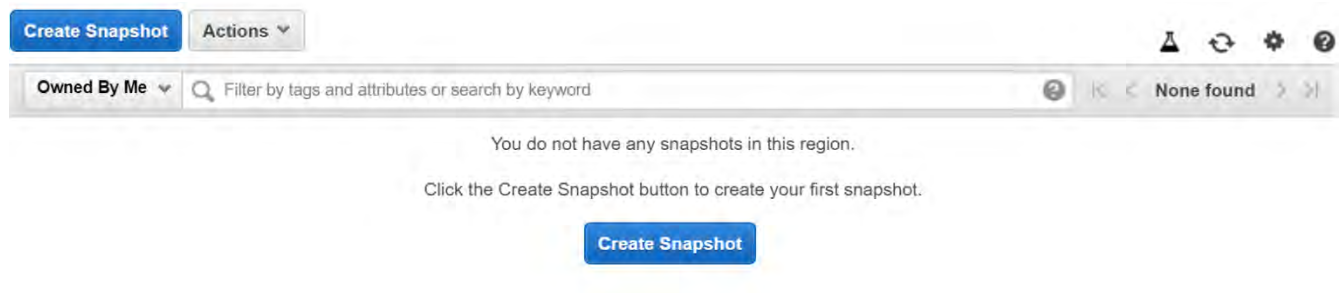
All RDS instances in spoke environments:

- *sampleapp-demo-env1*
- *sampleapp-demo-env2*
- *sampleapp-demo-env3*

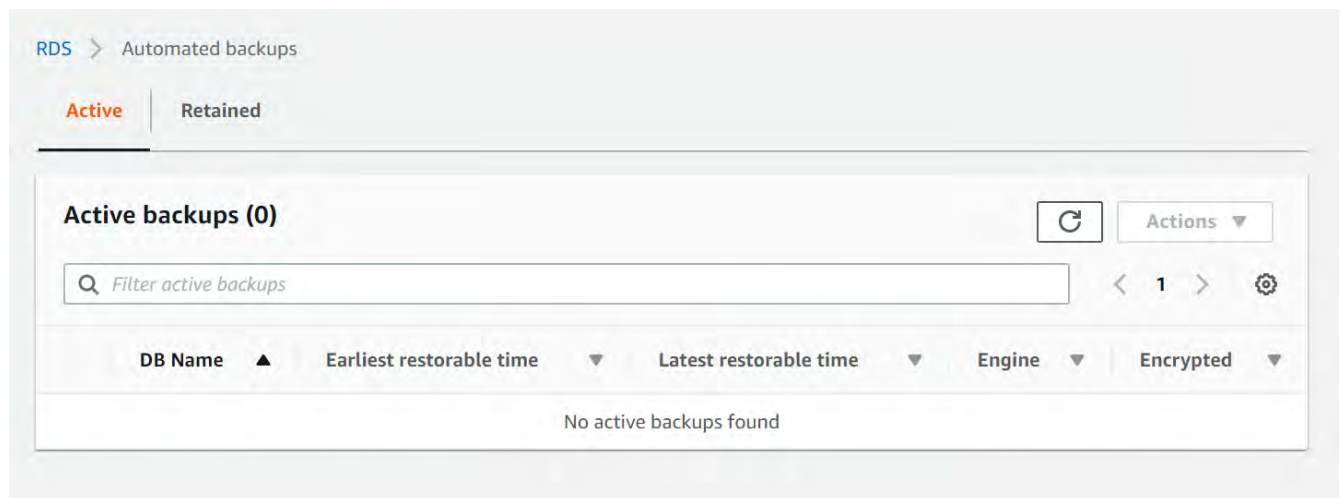
Gitlab's EBS volumes in *sampleapp-demo-devops* AWS account.

Evidences

An empty list of EBS snapshots in *sampleapp-demo- devops* AWS account:



An empty list of RDS automated backups in *sampleapp-demo-env3* AWS account:



Recommendations

DataArt recommends setting up basic EBS and [RDS backup strategies](#) in <Sample AWS> templates (like daily EBS snapshots with 7 days of retention and weekly automated RDS backups with 30 days of retention) and security controls for RDS backups and [EBS snapshots](#) (transparent encryption using AWS KMS), so the teams could build their own strategies using these reference configurations.

IN4. Misconfigured NACLs

Risk Rating: **INFO**

Summary

Network Access Control Lists (NACLs) usually provide an additional layer of security for cloud VPCs by acting like firewalls which control ingress and egress network traffic. In comparison with security groups (SGs), they support both allow and deny rules, and could be applied to a whole subnet, thus compensating rules in security groups which might be too permissive.

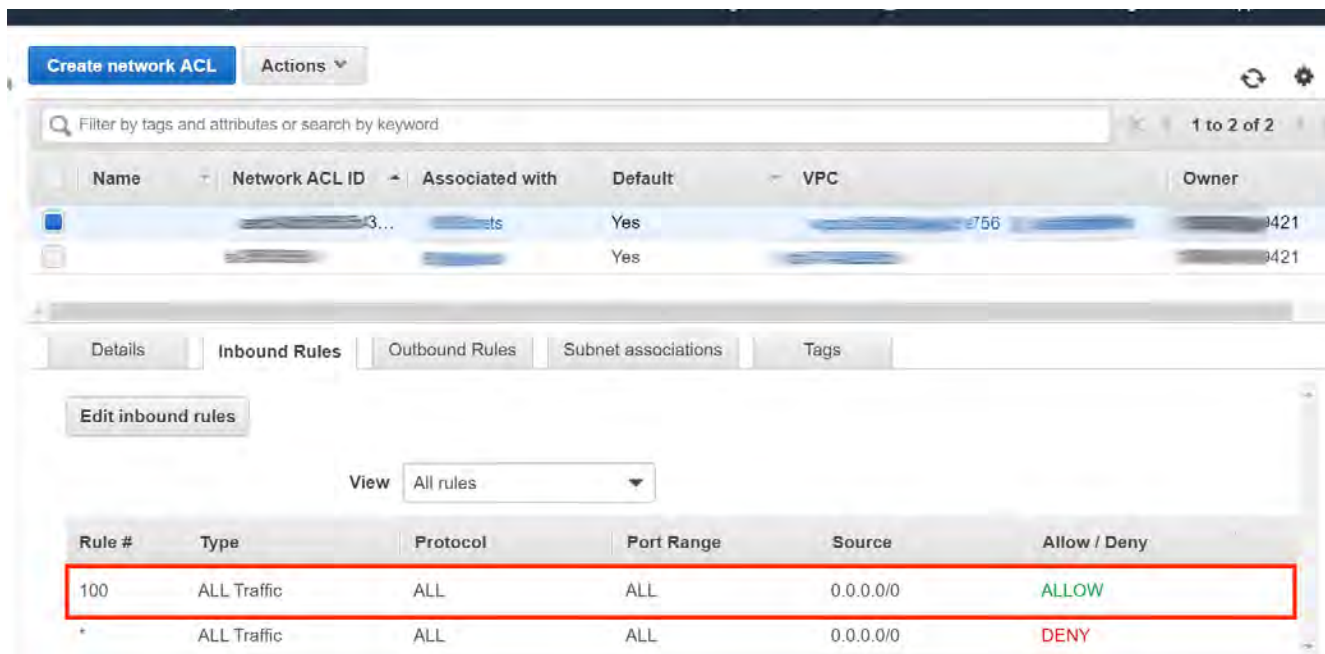
DataArt revealed that all NACLs used in <Sample AWS> environment used permissive ingress and egress rules, thus allowing traffic from all Internet addresses. Such implementation of network filtering is not compliant with two fundamental security principles – *Principle of Least Privilege* and *Security in Depth* (also known as Castle approach). The former assumes that a principal must be able to access only those resources that are necessary for its legitimate purpose; the latter requires the existence of multiple layers of security controls to significantly increase the time needed for a potential malefactor to access confidential data or gain full administrative access to the systems.

Affected Scope

All VPCs in all <Sample AWS> accounts

Evidences

Here's a picture of a NACL's inbound rule (Env1 account) – all traffic from all sources is allowed by default:



Recommendations

According to CIS Foundations Benchmark for AWS and Cloud Conformity KB, NACLs should be used to put extra restrictions on VPC level and compensate any potential flaws in security groups. Permissive rules in NACLs should be avoided unless there are strong reasons to do that.

IN5. Permissive rules for egress traffic in security groups

Risk Rating: INFO

Summary

While performing Terraform security code review, DataArt found out that multiple resources described in various modules defined fully opened egress security group rules. While this finding could not be classified as a real issue, such network configuration is not compliant with the principle of least privilege. Moreover, unrestricted outbound access could increase the risk of non-acceptable use of computational and network resources (like botnet or DDoS attack) in case a piece of infrastructure is compromised.

Affected Scope

All security groups in all <Sample AWS> accounts

Evidences

Here's a piece of *tfsec*'s output – there were multiple warnings related to fully open egress security group rules:

*[AWS007] Resource 'aws_security_group_rule.egress' defines a fully open egress security group rule.
tf-gitlab-runner /security.tf:32*

*[AWS007] Resource 'aws_security_group_rule.outbound_to_internet_http' defines a fully open egress security group rule.
tf-bastion/security.tf:32*

[AWS007] Resource 'aws_security_group_rule.outbound_to_internet_https' defines a fully open egress security group rule.
tf-bastion/security.tf:32

[AWS007] Resource 'aws_security_group_rule.bastion_egress_http' defines a fully open egress security group rule.
tf-gitlab/security.tf:39

[AWS007] Resource 'aws_security_group_rule.bastion_egress_https' defines a fully open egress security group rule.
tf-gitlab/security.tf:49

[AWS007] Resource 'aws_security_group_rule.consul_egress_https_access' defines a fully open egress security group rule.
tf-vault /network_security/consul.tf:43

[AWS007] Resource 'aws_security_group_rule.application_egress_http' defines a fully open egress security group rule.
tf-gitlab/security.tf:64

[AWS007] Resource 'aws_security_group_rule.application_egress_https' defines a fully open egress security group rule.
tf-gitlab/security.tf:74

[AWS007] Resource 'aws_security_group_rule.application_egress_smtp' defines a fully open egress security group rule.
tf-gitlab/security.tf:84

[AWS007] Resource 'aws_security_group_rule.tasks_to_world' defines a fully open egress security group rule.
tf-fargate-cluster/security_groups.tf:36

[AWS007] Resource 'aws_security_group_rule.runner_allow_out_all' defines a fully open egress security group rule.
tf-gitlab-tf-runner/security.tf:56

[AWS007] Resource 'aws_security_group_rule.outbound_internet_http' defines a fully open egress security group rule.

tf-postgres-rdbms/rds_security.tf:49

Recommendations

According to [Cloud Conformity recommendations](#), it's worth reviewing the list of potential external connections from EC2 instances and ECS containers and reducing the scope of allowed outgoing connections to be compliant with the principle of least privilege.

Conclusion

DataArt completed the security audit of <Sample AWS> configuration. This testing was based on the technologies and known threats as of the date of this document. All the security issues discovered during that exercise were analyzed and described in this report.

Please note that as technologies and risks change over time, the vulnerabilities associated with the operation of systems described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities, will also change.