Report for:

Agile Development Environment Security Assessment

Client Company Ltd.

September 2020

Version: 1.0

Prepared By:	TSC Consultant
Email:	t.consultant@nccgroup.com
Telephone:	+44 (0) 161 209 5200

NCC Group PLC - Security Testing Audit and Compliance

XYZ Building,
2 Hardman Boulevard,
Spinningfields,
Manchester,
M3 3AQ
https://www.nccgroup.com

Executive Summary

This report presents the findings of the Agile Development Environment Security Assessment conducted on behalf of Client Company Ltd. (Client). The assessment was conducted between 21/09/2020 and 30/09/2020 and was authorised by Client.

The system under assessment was an environment designed to support agile development and continuous software delivery. The assessment evaluated the security of this environment.

Overview

The security assessment identified a number of issues in relation to the systems within scope. The most significant issues were assessed to pose a high risk to Client and related to misconfigurations of key system components. It is recommended that the identified issues are addressed as described within this report in order to ensure that the organisation's information assets are suitably protected. This will, in turn, minimise the risk to which Client is exposed.

The following table breaks down the issues which were identified by phase and severity of risk (issues which are reported for information only are not included in the totals):

Phase	Description	Critical	High	Medium	Low	Total
1	Developer Scenario Assessment	0	1	1	0	**2**
2	Jenkins Review	0	2	2	1	**5**
3	AWS Configuration Review	0	0	9	1	**10**
4	Kubernetes Configuration	0	0	3	0	**3**
	Total	**0**	**3**	**15**	**2**	**20**

Assessment Summary

Jenkins and Rogue Developer Scenario Review

The most significant issues identified in the Jenkins review were an authorisation misconfiguration and a security weakness in an authorisation plugin. In combination, these could allow a registered user to gain access to the Jenkins instance and all existing jobs.

A global security setting granted read permissions to all authenticated users who were registered with GitHub, even if they were not a member of the organisation. The GitHub OAuth plugin had an existing vulnerability that ignored the permission relationships in place and made it possible to access the build output and console log of the jobs by any user.

The main concern identified by the rogue developer scenario assessment was that a number of credentials and an SSH private key were found in the GitHub repository. This could allow an attacker to gain access to multiple AWS S3 buckets and download sensitive data.

AWS Review

The most significant issue identified in the AWS review was the use of IAM roles which were overly permissive. This could allow a trusted user to elevate their privileges within the account in order to perform actions which they should not be able to perform.

It was also observed that the password policy applied to IAM users was weaker than recommended. Weak password policies typically lead to the creation and use of weak passwords which are more susceptible to recovery by brute-force and password guessing attacks. It is acknowledged, however that this risk was mitigated by the use of multi-factor authentication (MFA) across all accounts.

A number of other issues where the configuration departed from security best practice were noted, although none were assessed to pose more than a medium risk. For example, two issues were noted with regard to the configuration of the S3 buckets. A number of buckets did not encrypt the data they stored while others used a clear text channel for their communications. It should be ensured that any potentially sensitive data is encrypted both when at rest and as it is communicated.

Kubernetes Review

Only a small number of issues were identified in the Kubernetes review. The most significant was that the version of Kubernetes in use within the cluster was outdated and so contained a number of security vulnerabilities. While the vulnerable features were not in use, this still illustrates the importance of keeping the installed version of Kubernetes up to date and so minimising the risk of becoming vulnerable to newly discovered issues.

In addition, the pod security policy and cluster networking rules would benefit from review in order to bring the Kubernetes environment into line with security best practice.

The remaining issues were all assessed to pose a low risk or are reported for information only. Nevertheless, it is recommended that they are all reviewed and addressed so as to bring the environment within scope into line with security best practice. It is important to recognise that even low risk issues can be exploited in combination with other issues as part of a wider attack which seeks to compromise an environment or application. In addition, resolving lower risk issues can have the dual benefit of reducing the attractiveness of systems to opportunistic attackers as well as enhancing the overall security posture.

More detailed information on each of the issues which were identified is included in Section 2 of this report.

Strategic Recommendations

The majority of the identified issues were the result of hosts or systems that did not make the most of the secure configurations available. These should be reconfigured in line with the security best practice guidance provided later in this report.

In particular, it is recommended that, in addition to removing any of the insecurely stored credentials which were identified, steps should be taken to ensure that no further secrets are stored in GitHub repositories. Consideration should also be given to implementing a periodic review of all GitHub repositories for strings of characters which could indicate that secrets have been exposed.

Any remedial actions which are undertaken as a result of this assessment should also be reviewed for inclusion in the organisation’s secure build standards and deployment procedures.

It is recommended that the issues set out in this report should be addressed by a structured programme of remedial actions which are prioritised in accordance with the perceived risk to the organisation.

Table of Contents

Using This Report

To facilitate the dissemination of the information within this report throughout your organisation, this document has been divided into the following clearly marked and separable sections.

Document Breakdown		
	Executive Summary	Management level, strategic overview of the assessment and the risks posed to the business
1	Technical Summary	An overview of the assessment from a more technical perspective, including a defined scope and any caveats which may apply
2	Technical Details	Detailed discussion (including evidence and recommendations) for each individual security issue which was identified
3	Supplemental Data	Any additional evidence which was too lengthy to include in Section 2
4	Appendices	This section usually includes the security tools which were used, outlines the assessment methodologies and lists the assessment team members

Document Control

Client Confidentiality

This document contains Client Confidential information and may not be copied without written permission.

Proprietary Information

The content of this document should be considered proprietary information and should not be disclosed outside of Client Company Ltd..

NCC Group gives permission to copy this report for the purposes of disseminating information within your organisation or any regulatory agency.

Document Version Control	
Data Classification	Client Confidential
Client Name	Client Company Ltd.
Project Reference	9999999
Proposal Reference	O-12345
Document Title	Agile Development Environment Security Assessment
Author	TSC Consultant

Document History			
Issue No.	**Issue Date**	**Issued By**	**Change Description**
0.1	29/09/2020	TSC Consultant	Draft for NCC Group internal review only
0.2	01/10/2020	A Reviewer	Revised QA
1.0	02/10/2020	TSC Consultant	Released to client

Document Distribution List	
A. Sponsor	Project Sponsor, Client
TSC Consultant	Senior Security Consultant, NCC Group
A. Manager	Account Manager, NCC Group

1 Technical Summary

NCC Group was contracted by Client to conduct a security assessment of the systems within scope in order to identify security issues that could negatively affect Client's business or reputation if they led to the compromise or abuse of systems.

1.1 Scope

The security assessment was carried out in the staging environment and included:

- Security Evaluation of a Rogue Developer Scenario
- Jenkins Review
- AWS Configuration Review
- Kubernetes Review

The URL and GitHub within the scope of the Jenkins and Rogue Developer Scenario are listed below:

- https://project-app-jenkins.staging.clientuk
- https://github/clientuk/project

The following AWS account was reviewed:

- 123456789012

The following Kubernetes cluster was reviewed:

- project-staging

1.2 Caveats

Due to the nature of the environment, checks that would have a high probability of causing disruption to the named hosts were excluded. Denial of service attempts were excluded for the same reason.

1.3 Post Assessment Cleanup

The following test accounts which were created for the purpose of this assessment should be disabled or removed, as appropriate, together with any associated content:

- testuser1
- testuser2
- testadmin1
- testadmin2

Revert any WAF/IDS/IPS/firewall changes which were made for the purposes of the assessment.

1.4 Risk Ratings

The table below gives a key to the icons and symbols used throughout this report to provide a clear and concise risk scoring system.

It should be stressed that quantifying the overall business risk posed by any of the issues found in any test is outside our remit. This means that some risks may be reported as high from a technical perspective but may, as a result of other controls unknown to us, be considered acceptable.

Symbol	Risk Rating	CVSSv2 Score	Explanation
	CRITICAL	9.0 - 10	A vulnerability was discovered that has been rated as critical. This requires resolution as quickly as possible.
	HIGH	7.0 - 8.9	A vulnerability was discovered that has been rated as high. This requires resolution in the short term.
	MEDIUM	4.0 - 6.9	A vulnerability was discovered that has been rated as medium. This should be resolved as part of the ongoing security maintenance of the system.
	LOW	1.0 - 3.9	A vulnerability was discovered that has been rated as low. This should be addressed as part of routine maintenance tasks.
	INFO	0 - 0.9	A discovery was made that is reported for information. This should be addressed in order to meet leading practice.
	N/A	N/A	Good security practices were being followed or an audit item was found to be present and correct.

1.5 Findings Overview

All the issues identified during the assessment are listed below with a brief description and risk rating for each issue. The risk ratings used in this report are defined in Section 1.4 Risk Ratings.

Phase 1 – Developer Scenario Assessment

Ref	Finding	Risk
SR-128-1-1	***SSH Private Key in S3 Bucket*** One SSL private key was identified in the private GitHub repository. Whilst the key was not immediately exposed to unauthorised access, storing keys such as this in an online location (and in an unencrypted format) places them at risk of compromise if an attacker gains access to the GitHub account or AWS S3 bucket. Information could also be disclosed if a client device where the repositories had been cloned were lost or stolen.	High 
SR-128-1-2	***Credentials in GitHub Repository*** Some instances of what appeared to be AWS and JFrog passwords or credentials were noted in the private GitHub Project repository. Whilst it was not possible to establish from the repositories alone whether all of these were in use in production, in general it is not recommended that credentials should be stored in source code repositories in clear text.	Medium 
SR-128-1-3	***Good Practice Observed*** It was apparent that security had been a consideration in the development and deployment of the pipeline and its environment. The risk from various security flaws had been effectively mitigated. A number of useful controls and processes were observed to be in place.	Good 

Phase 2 – Jenkins Review

Ref	Finding	Risk
SR-128-2-1	***GitHub OAuth Misconfiguration*** The GitHub OAuth Plugin provided authentication and authorisation methods for the Jenkins server. The authorisation settings granted read permissions to all Authenticated Users. This meant that any user registered with and logged into GitHub could access Jenkins, even though the Participant in Organization setting was originally set to prevent users from outside the organisation being able to access Jenkins.	High 
SR-128-2-2	***Jenkins Broken Access Control*** Jenkins extends its functionality with plugins that can be developed by third party companies or by individuals. The GitHub OAuth plugin was identified as being responsible for authentication and authorisation. However, this plugin contained a bug that ignored the permission relationships and allowed any users to access data, even when this should have been denied as a result of that user's permissions.	High 
SR-128-2-3	***Outdated Jenkins Software*** The installed version of Jenkins Server was outdated and affected by a number of security vulnerabilities. The observed version of the Jenkins was 2.231 while the latest version (at the time of writing) is 2.259. Known software vulnerabilities which were unpatched in the observed version included CSRF and XSS weaknesses. Known issues also included missing permissions checks and the potential disclosure of SMTP passwords.	Medium 
SR-128-2-4	***CSRF Protection Disabled*** The built-in Jenkins CSRF token had not been enabled. This increases the risk of a successful CSRF attack which could allow an attacker to manipulate a user or an administrator into unwittingly performing actions within the application (such as changing the account password, adding accounts to the system and changing settings) on behalf of the attacker.	Medium 
SR-128-2-5	***Clear Text Credentials*** Jenkins uses the workspace directory to build projects, to store the checkout source code and to store the files generated by the build itself. However, some files in the workspace directory were not automatically deleted. Some of these remaining files included credentials in clear text format, and had world readable permissions.	Low 

Phase 3 – AWS Configuration Review

Ref	Finding	Risk
SR-128-3-1	***Root Account Used Recently*** It was confirmed that the "root" account had been used recently as part of the security checks of the Amazon console. Amazon security best practice recommends that this account should only be used when strictly necessary. Other accounts with more suitably restricted permissions should be used for daily administrative duties.	Medium
SR-128-3-2	***IAM Password Policy*** Weaknesses were noted in the password policy enabled in the IAM settings. This could result in the creation of weak user passwords, more easily susceptible to recovery by an attacker. Furthermore, these settings could allow a weak password to be used for an extended period of time.	Medium
SR-128-3-3	***EBS Volumes Without Encryption*** Elastic Block Store (EBS) based full disk encryption (FDE) was not enabled on the EBS volumes. Due to the nature of a cloud-hosted environment, hosted data stored on virtual drives can be re-assigned to other hosts, often beyond the account that owns it. If an administrator removes a virtual drive from a hosted machine, this data can be re-assigned elsewhere and in some circumstances recovered.	Medium
SR-128-3-4	***Automated Backups Not Enabled for RDS Instances*** Automated backups had not been enabled for a number of Relational Database Service (RDS) instances. This feature facilitates the creation of periodic RDS instance snapshots, which in turn ensure that data restoration is possible in the event of an incident affecting the source database.	Medium
SR-128-3-5	***S3 Bucket Configuration Allowing Clear Text Communications*** The observed configuration for several S3 buckets did not enforce the use of SSL/TLS connections, or prevent clients from connecting over clear text protocols. Communication between clients and S3 buckets could use unencrypted HTTP as well as HTTPS as the use of HTTPS was not enforced. As a result, sensitive information could be transmitted in clear text over the network.	Medium
SR-128-3-6	***S3 Bucket Without Default Encryption*** Analysis of the S3 configuration indicated that a number of buckets did not have default encryption enabled. S3 default encryption provides a way to set the default encryption behaviour for an S3 bucket. When enabled, all objects will be encrypted when they are stored in the bucket. This will be done automatically, without having to construct a bucket policy that rejects objects that are not encrypted.	Medium
SR-128-3-7	***Policy Allows "AssumeRole" Action For Any Resource*** One of the IAM Inline security policies reviewed during the assessment included the AssumeRole action, without limiting it to a specific resource. The Security Token Service (STS) is a service that enables requesting temporary, limited-privilege credentials for users. The AssumeRole action implemented by the STS returns a set of temporary credentials that can be used to access AWS resources. A user, group or role with access to this action through the identified policies would be allowed to assume any role in any account that trusts the user's account. This is contrary to the principle of least privilege, as suggested by AWS security best practice.	Medium

Ref	Finding	Risk
SR-128-3-8	***Policy Allows "PassRole" Action For Any Resource*** A number of the IAM Managed and Inline security policies reviewed during the assessment included the "PassRole" action, without limiting it to a specific resource. To pass a role (and its permissions) to an AWS service, a user must have permissions to pass the role to the service. This helps administrators ensure that only approved users can configure a service with a role that grants permissions. Creating a policy that includes the "PassRole" permission without restrictions could enable the entity to which this policy is attached to access resources to which it has not been granted explicit access. This is contrary to the principle of least privilege, as suggested by AWS security best practice.	Medium 
SR-128-3-9	***Default Security Group Hardening Not Implemented*** A number of security groups were identified to be using the default rules, allowing all inbound traffic from instances assigned to the same security group, as well as all outbound traffic. The default security group is assigned to new instances created within a VPC if no custom security groups are assigned to it during configuration. These rules may be overly permissive, for instance allowing an attacker who has compromised one instance with the default security group assigned to use horizontal privilege escalation to compromise all other instances configured with the default security group.	Medium 
SR-128-3-10	***Single Availability Zone RDS Instance*** One Relational Database Service (RDS) instance was configured without the Multi Availability Zone option enabled. Without this, should an availability zone specific database failure occur, then Amazon RDS cannot automatically fail over to the standby availability zone so that database operations can resume quickly without administrative intervention.	Low 

Phase 4 – Kubernetes Configuration

Ref	Finding	Risk
SR-128-4-3	***Outdated Kubernetes in Use*** The Kubernetes service was found to be out of date, leaving the clusters and their respective containers potentially open to both security and functional issues that have been patched in later versions.	Medium 
SR-128-4-1	***Kubernetes Pod Security Policy Overly Permissive*** The cluster did make use of the Kubernetes Pod Security Policy feature, but was configured to allow any user to create privileged containers. As a result, any user with the ability to create new pods on the cluster would be able to gain privileged access to the underlying cluster nodes and any secrets contained within them.	Medium 
SR-128-4-2	***Kubernetes Unrestricted Cluster Network*** There were no egress restrictions on traffic from pods in the cluster network. As a result, a compromised container would be able to attack Kubernetes' management interfaces.	Medium 

2 Technical Details

The remainder of this document is technical in nature and provides additional detail about the items already discussed, for the purposes of remediation and risk assessment.

2.1 Detailed Findings

2.1.1 Phase 1 – Developer Scenario Assessment

SR-128-1-1	*SSH Private Key in S3 Bucket*	
Risk Rating	High	

Description:

One SSL private key was identified in the private GitHub repository. Whilst the key was not immediately exposed to unauthorised access, storing keys such as this in an online location (and in an unencrypted format) places them at risk of compromise if an attacker gains access to the GitHub account or AWS S3 bucket. Information could also be disclosed if a client device where the repositories had been cloned were lost or stolen.

Recommendation:

Ensure that all SSL private keys which will be used in test, staging and production environments are stored securely and not exposed on public facing systems such as a GitHub repository or in AWS S3 buckets.

Affects:

S3 Bucket Name
s3://dev.client.com

SR-128-1-2	*Credentials in GitHub Repository*	
Risk Rating	Medium	

Some instances of what appeared to be AWS and JFrog passwords or credentials were noted in the private GitHub Project repository. Whilst it was not possible to establish from the repositories alone whether all of these were in use in production, in general it is not recommended that credentials should be stored in source code repositories in clear text.

Instances of passwords stored in cloud formation template files:

```
https://github/clientuk/project/develop/config
https://github/clientuk/project/develop/deployment/fullstart.json
```

Additionally, a hard-coded password was noted in Python and shell scripts:

```
https://github/clientuk/project/develop/scripts/s3_pull.py
https://github/clientuk/project/develop/deployment/s3_pull_t.py
```

It was not possible to perform any successful privilege escalation with these credentials. The AWS credentials were for roles or IAM users with S3 buckets who were accorded only limited permissions.

A list of the files found to contain passwords is available in Supplemental Data, Section 3.1.

Recommendation:

Ensure that credentials are not stored in clear text in GitHub repositories or in AWS S3 buckets.

Affects:

GitHub Repository
Project

SR-128-1-3	*Good Practice Observed*	
Risk Rating	Good	

Description:

It was apparent that security had been a consideration in the development and deployment of the pipeline and its environment. The risk from various security flaws had been effectively mitigated. A number of useful controls and processes were observed to be in place.

These included:

- Enforced code reviews
- Defined checks for committing
- Separate teams were responsible for different systems
- Developers could not bypass pipeline steps to introduce new code or change existing code
- The use of multi-factor authentication was enforced for all AWS and GitHub access

Recommendation:

Ensure that these practices are maintained by means of training programmes and supporting procedures which define secure and acceptable practices within the organisation.

2.1.2 Phase 2 – Jenkins Review

SR-128-2-1	*GitHub OAuth Misconfiguration*	
Risk Rating	High	

Description:

The GitHub OAuth Plugin provided authentication and authorisation methods for the Jenkins server. The authorisation settings granted read permissions to all Authenticated Users. This meant that any user registered with and logged into GitHub could access Jenkins, even though the Participant in Organization setting was originally set to prevent users from outside the organisation being able to access Jenkins.

Figure 1 –READ permissions granted to all Authenticated Users

It was possible to exploit this issue in combination with the security weakness described in the Jenkins Broken Access Control issue in order to gain access to the build output from all jobs.

Figure 2 - Accessing Jenkins with a user who was not a member of the Organization

Recommendation:

Remove the read permissions to all Authenticated Users in the Global Security Settings.

Affects:

DNS Name
project-app-jenkins.staging.clientuk

References:

GitHub OAuth Plugin

https://wiki.jenkins.io/display/JENKINS/GitHub+OAuth+Plugin

SR-128-2-2	*Jenkins Broken Access Control*	
Risk Rating	High	

Description:

Jenkins extends its functionality with plugins that can be developed by third party companies or by individuals. The GitHub OAuth plugin was identified as being responsible for authentication and authorisation. However, this plugin contained a bug that ignored the permission relationships and allowed any users to access data, even when this should have been denied as a result of that user's permissions.

Figure 3 – Current user had no access to job details by default

Figure 4 – Bypassing restriction using the available third party URL

Recommendation:

Apply the workaround in the References or consider using a different authorisation and authentication plugin (after testing has provided a suitable level of assurance).

Affects:

DNS Name
project-app-jenkins.staging.clientuk

References:

GitHub OAuth Plugin Bug

https://issues.jenkins-ci.org/browse/JENKINS-54031

SR-128-2-3	*Outdated Jenkins Software*	
Risk Rating	Medium	

Description:

The installed version of Jenkins Server was outdated and affected by a number of security vulnerabilities. The observed version of the Jenkins was 2.231 while the latest version (at the time of writing) is 2.259. Known software vulnerabilities which were unpatched in the observed version included CSRF and XSS weaknesses. Known issues also included missing permissions checks and the potential disclosure of SMTP passwords.

These vulnerabilities could allow an attacker to steal the session of an administrator, gain access to sensitive information or create files and potentially execute code on the host system. This issue has been rated as a medium risk to reflect these issues and their assigned CVSS scores.

Recommendation:

Update the version of the affected Jenkins server to the latest stable and secure version available. Perform any testing necessary to ensure that this does not break or conflict with required functionality.

Review the current policy for upgrading Jenkins servers and create a plan that ensures Jenkins can be maintained at the latest stable version, while still ensuring adequate testing so as to minimise disruption.

Affects:

DNS Name
project-app-jenkins.staging.clientuk

References:

Jenkins Security Advisory 2020-08-12

https://www.jenkins.io/security/advisory/2020-08-12/

Jenkins Website: ChangeLog

https://jenkins.io/changelog/

https://jenkins.io/changelog-stable/

SR-128-2-4	*CSRF Protection Disabled*	
Risk Rating	Medium	

Description:

The built-in Jenkins CSRF token had not been enabled. This increases the risk of a successful CSRF attack which could allow an attacker to manipulate a user or an administrator into unwittingly performing actions within the application (such as changing the account password, adding accounts to the system and changing settings) on behalf of the attacker.

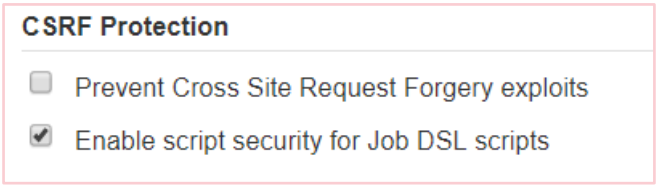

Figure 5 – CRSF protection option not enabled

For a CSRF attack to succeed, the victim must click a malicious link or visit a malicious web page while logged in to the vulnerable application. This would cause the victim's browser to issue GET or POST requests to the vulnerable site by means of hidden images or embedded JavaScript code. As far as the application is concerned, these forged requests would appear to be the result of genuine actions performed by real users clicking on elements within the application. This is because the application did not sufficiently validate requests (such as by means of random tokens) to allow it to make this distinction.

Recommendation:

Enable the CSRF token protection in Jenkins.

One time tokens are the primary defence against CSRF. Tokens are validated server-side to ensure that requests can only be executed once. The token which is received must match the server-side information.

Affects:

DNS Name
project-app-jenkins.staging.clientuk

References:

Jenkins CSRF Protection

https://wiki.jenkins.io/display/JENKINS/CSRF+Protection

Remote Access API

https://wiki.jenkins.io/display/JENKINS/Remote+access+API

Jenkins CSRF Protection Explained

https://support.cloudbees.com/hc/en-us/articles/219257077-CSRF-Protection-Explained

SR-128-2-5	*Clear Text Credentials*	
Risk Rating	Low	

Description:

Jenkins uses the workspace directory to build projects, to store the checkout source code and to store the files generated by the build itself. However, some files in the workspace directory were not automatically deleted. Some of these remaining files included credentials in clear text format, and had world readable permissions.

Figure 6 – Example file containing API key in the workspace directory

Figure 7 – Some files had world readable permissions in the workspace directory

This could allow an attacker with SSH or script console access to recover credentials to different systems.

Recommendation:

Use the Workspace Cleanup plugin to remove files that contain credentials such as Docker configuration files or shell scripts.

Affects:

DNS Name
project-app-jenkins.staging.clientuk

References:

Workspace Cleanup Plugin

https://wiki.jenkins.io/display/JENKINS/Workspace+Cleanup+Plugin

How to use Workspace Cleanup Plugin

https://blog.clairvoyantsoft.com/jenkins-cleanup-workspace-reduce-disk-usage-18310097d3ef

2.1.3 Phase 3 – AWS Configuration Review

SR-128-3-1	*Root Account Used Recently*	
Risk Rating	Medium	

Description:

It was confirmed that the “root” account had been used recently as part of the security checks of the Amazon console. Amazon security best practice recommends that this account should only be used when strictly necessary. Other accounts with more suitably restricted permissions should be used for daily administrative duties.

The following screenshot demonstrates that the root account was used recently on the 2020-08-12:

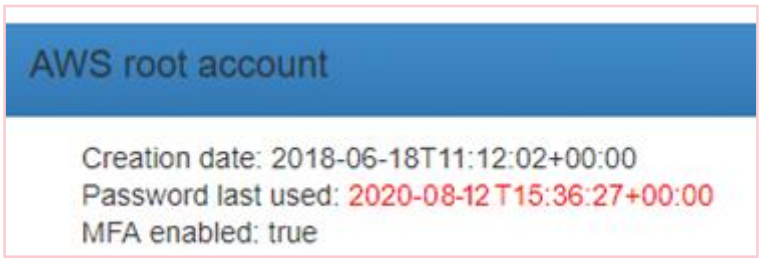

Figure 8 – Root account usage

The use of the root account was discussed with Client staff, who revealed that the use of the root account is part of a corporate account, and project staff had no direct control over the account.

Recommendation:

The use of the root account should be minimised. Other accounts should be created following the principle of least privilege and used for management. This will reduce the risk of accidental changes and unintended disclosure of highly privileged credentials.

This would also better align the AWS account with Amazon recommended security best practice.

Affects:

AWS Account
123456789012:root

References:

AWS Identity and Access Management (IAM)

https://aws.amazon.com/iam/

AWS Identity and Access Management Documentation

https://aws.amazon.com/documentation/iam/

AWS Documentation – IAM Best Practices

http://docs.aws.amazon.com/IAM/latest/UserGuide/IAMBestPractices.html

CIS Amazon Web Services Foundations

https://d0.awsstatic.com/whitepapers/compliance/AWS_CIS_Foundations_Benchmark.pdf

SR-128-3-2	*IAM Password Policy*	
Risk Rating	Medium	

Description:

Weaknesses were noted in the password policy enabled in the IAM settings. This could result in the creation of weak user passwords, more easily susceptible to recovery by an attacker. Furthermore, these settings could allow a weak password to be used for an extended period of time.

The following screenshot illustrates the password policy settings that were observed. Note that the check boxes for the options `Prevent password reuse`, `Require at least one non-alphanumeric character` and `Enable password expiration` were not selected:

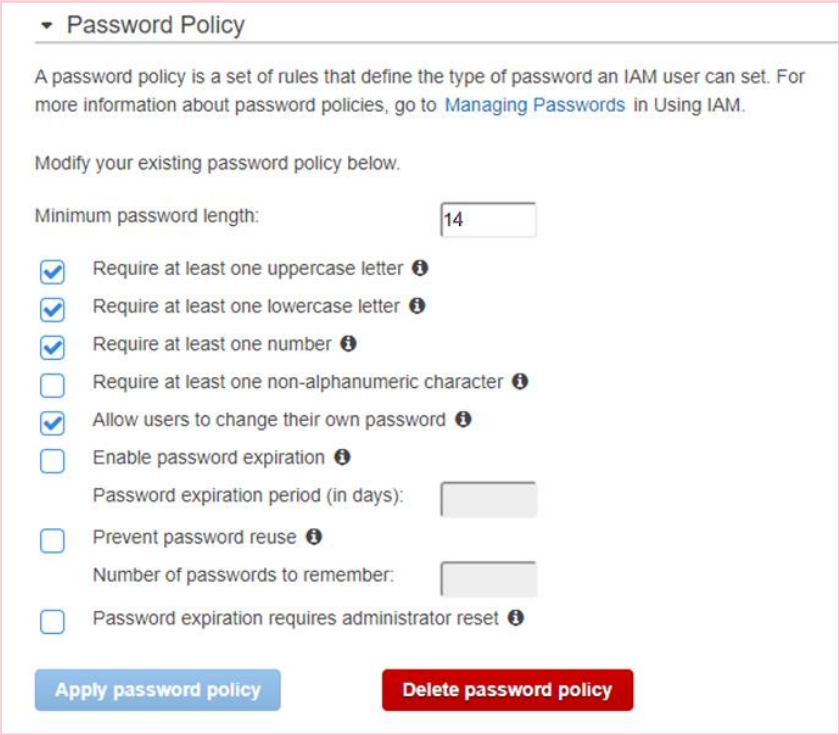

Figure 9 – Password policy not enforced

The `Prevent password reuse` option will not allow a user to reuse a previous password value when setting a new password (which otherwise might allow continued use of the same password). The number of previous passwords which are not allowed can be set between 1 and 24.

Passwords can be set to expire after a period between 1 and 1095 days. It is worth noting that some more recent password guidance suggests *not* setting passwords to expire, but that this advice is given in the expectation that all passwords are of suitably strong and that other controls (such as MFA) are used where appropriate.

Recommendation:

Ensure a strong password policy is in place. Specifically, ensure that the following password policy is enforced:

- **Passwords should include at least one symbol**
- **Passwords cannot be reused**
- **Passwords should be reused**
- Passwords should use mixed case
- Passwords should include at least one number
- Passwords require minimum length of 14 characters

Affects:

AWS Account
123456789012

References:

AWS Identity and Access Management (IAM)

https://aws.amazon.com/iam/

AWS Identity and Access Management Documentation

https://aws.amazon.com/documentation/iam/

AWS Documentation – Managing Passwords

http://docs.aws.amazon.com/IAM/latest/UserGuide/Credentials-ManagingPasswords.html

SR-128-3-3	*EBS Volumes Without Encryption*	
Risk Rating	Medium	

Description:

Elastic Block Store (EBS) based full disk encryption (FDE) was not enabled on the EBS volumes. Due to the nature of a cloud-hosted environment, hosted data stored on virtual drives can be re-assigned to other hosts, often beyond the account that owns it. If an administrator removes a virtual drive from a hosted machine, this data can be re-assigned elsewhere and in some circumstances recovered.

Amazon have stated that drives are wiped after deletion; however, FDE ensures that if a virtual drive is removed and re-assigned to a different host controlled by another AWS account then this data would be very difficult, if not impossible, to recover.

The following figure highlights the volumes that did not have encryption enabled:

Figure 10 – EBS volumes lacking encryption

114 volumes were flagged and are detailed in Supplemental Data, Section 3.3.

Recommendation:

Encrypt all hosted machine virtual drives that host intellectual property or sensitive data.

Furthermore, OS based encryption (such as LUKS or BitLocker) can be leveraged in order to mitigate the risk from this issue.

Affects:

See Supplemental Data, Section 3.3.

References:

Amazon EC2

https://aws.amazon.com/ec2/

Amazon Elastic Compute Cloud Documentation

https://aws.amazon.com/documentation/ec2/

AWS Security Whitepaper – Elastic Block Storage Security (page 24)

https://d0.awsstatic.com/whitepapers/aws-security-whitepaper.pdf

AWS Documentation – Amazon Elastic Block Store (Amazon EBS)

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AmazonEBS.html

SR-128-3-4	*Automated Backups Not Enabled for RDS Instances*	
Risk Rating	Medium	

Automated backups had not been enabled for a number of Relational Database Service (RDS) instances. This feature facilitates the creation of periodic RDS instance snapshots, which in turn ensure that data restoration is possible in the event of an incident affecting the source database.

The following screenshot from the AWS Trusted Advisor shows that the `main-components-db-stage` and `region-components-db-stage` RDS instances had automated backups disabled:

Figure 11 – RDS Instances with automated backups disabled

Recommendation:

Ensure that all RDS database instances that host sensitive data have automated backups enabled for point-in-time recovery.

Affects:

RDS Instance
main-components-db-stage
region-components-db-stage

References:

Amazon Relational Database Service (RDS)

https://aws.amazon.com/rds/

Amazon Relational Database Service Documentation

https://aws.amazon.com/documentation/rds/

AWS Documentation – Backing Up and Restoring Amazon RDS DB Instances

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_CommonTasks.BackupRestore.html

SR-128-3-5	*S3 Bucket Configuration Allowing Clear Text Communications*	
Risk Rating	Medium	

Description:

The observed configuration for several S3 buckets did not enforce the use of SSL/TLS connections, or prevent clients from connecting over clear text protocols. Communication between clients and S3 buckets could use unencrypted HTTP as well as HTTPS as the use of HTTPS was not enforced. As a result, sensitive information could be transmitted in clear text over the network.

The following buckets were found not to enforce the use of encrypted communication:

- project-jenkins-stage
- project-xa-jenkins-stage
- project-xb-jenkins-stage

The following screenshot shows the policy for one of the buckets listed above:

REDACTED

Figure 12 – Policy allowing clear text communications

This configuration presents an opportunity for an intercepting party to capture sensitive data. In general, any data that is in any way sensitive should only be communicated over HTTPS.

Recommendation:

The use of clear text protocols should be prevented by enforcing TLS as a transport method.

Adding the following statement to a bucket configuration will ensure objects can only be accessed over TLS connections.

```
{
    "Statement":[
        {
            "Action": "s3:*",
            "Effect":"Deny",
            "Principal": "*",
            "Resource":["arn:aws:s3:::<bucketname>",
                        "arn:aws:s3:::<bucketname>/*"],
            "Condition":{
                "Bool":
                { "aws:SecureTransport": false }
            }
        }
    ]
}
```

Affects:

S3 Buckets
project-jenkins-stage
project-xa-jenkins-stage
project-xb-jenkins-stage

References:

Amazon S3

https://aws.amazon.com/s3/

Amazon Simple Storage Service Documentation

https://aws.amazon.com/documentation/s3/

AWS Documentation – AWS Global and IAM Condition Context Keys

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html

AWS Documentation – IAM JSON Policy Elements: Condition Operators

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_condition_operators.html

AWS Documentation – IAM Policy Elements: Variables

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html

SR-128-3-6	*S3 Bucket Without Default Encryption*	
Risk Rating	Medium	

Description:

Analysis of the S3 configuration indicated that a number of buckets did not have default encryption enabled. S3 default encryption provides a way to set the default encryption behaviour for an S3 bucket. When enabled, all objects will be encrypted when they are stored in the bucket. This will be done automatically, without having to construct a bucket policy that rejects objects that are not encrypted.

The following buckets were found not to have default encryption enabled:

- project-jenkins-stage
- project-xa-jenkins-stage
- project-xc-jenkins-stage
- project-y99-jenkins-stage
- project-za-jenkins-stage
- project-xab-jenkins-stage
- project-zd-jenkins-stage

The following screenshot shows an S3 bucket with default encryption not enabled:

REDACTED

Figure 13 – Bucket without default encryption enabled

It was noted that default encryption *was* enabled on other S3 buckets configured on the system.

Recommendation:

Enable default encryption on all S3 buckets.

Affects:

S3 Buckets
project-jenkins-stage
project-xa-jenkins-stage
project-xc-jenkins-stage
project-y99-jenkins-stage
project-za-jenkins-stage
project-xab-jenkins-stage
project-zd-jenkins-stage

References:

Amazon S3

https://aws.amazon.com/s3/

Amazon Simple Storage Service Documentation

https://aws.amazon.com/documentation/s3/

AWS Documentation – Using Encryption

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

AWS Documentation – How Do I Create an S3 Bucket?

https://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-bucket.html

AWS Documentation – Amazon S3 Default Encryption for S3 Buckets

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

AWS News Blog - New Amazon S3 Encryption & Security Features

https://aws.amazon.com/blogs/aws/new-amazon-s3-encryption-security-features/

SR-128-3-7	*Policy Allows "AssumeRole" Action For Any Resource*	
Risk Rating	Medium	

Description:

One of the IAM Inline security policies reviewed during the assessment included the `AssumeRole` action, without limiting it to a specific resource. The Security Token Service (STS) is a service that enables requesting temporary, limited-privilege credentials for users. The `AssumeRole` action implemented by the STS returns a set of temporary credentials that can be used to access AWS resources. A user, group or role with access to this action through the identified policies would be allowed to assume any role in any account that trusts the user's account. This is contrary to the principle of least privilege, as suggested by AWS security best practice.

The following security policy was affected by this issue:

- arn:aws:iam::123456789012:policy/jenkins-policy

The following statement was included in this policy:

```
{
  "Action": [
    "sts:AssumeRole"
  ],
  "Effect": "Allow",
  "Resource": [
    "*"
  ]
}
```

If `Resource` is set to `*` as in the case above, then the user receiving this policy can assume any role in any account that trusts the user's AWS account (that is, the role's trust policy specifies the user's account as Principal).

Recommendation:

The policies affected by this issue should be reviewed.

One potential mitigation is to specify the complete ARN of the role that the user should be allowed to assume. For example, the following policy sets this restriction:

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::<account-id>:role/[role to be assumed]"
  }
}
```

This helps prevent the user from assuming roles that have more permissions than the user has been granted - that is, from being able to obtain elevated privileges.

Affects:

IAM Policy
jenkins-policy

References:

AWS Identity and Access Management (IAM)

https://aws.amazon.com/iam/

AWS Identity and Access Management Documentation

https://aws.amazon.com/documentation/iam/

AWS Documentation – Granting a User Permissions to Switch Roles

http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_permissions-to-switch.html

AWS Documentation – AssumeRole

https://docs.aws.amazon.com/STS/latest/APIReference/API_AssumeRole.html

SR-128-3-8	*Policy Allows "PassRole" Action For Any Resource*	
Risk Rating	Medium	

Description:

A number of the IAM Managed and Inline security policies reviewed during the assessment included the "PassRole" action, without limiting it to a specific resource. To pass a role (and its permissions) to an AWS service, a user must have permissions to pass the role to the service. This helps administrators ensure that only approved users can configure a service with a role that grants permissions. Creating a policy that includes the "PassRole" permission without restrictions could enable the entity to which this policy is attached to access resources to which it has not been granted explicit access. This is contrary to the principle of least privilege, as suggested by AWS security best practice.

The following security policies were affected by this issue:

- arn:aws:iam::123456789012:role/jenkins-role
- arn:aws:iam::123456789012:role/jenkins-role1

The following "PassRole" permission was configured for the `jenkins-role` policy:

```
{
  "Action": [
    "iam:AddRoleToInstanceProfile",
    "iam:AttachRolePolicy",
    "iam:CreateInstanceProfile",
    "iam:CreatePolicy",
    "iam:CreateRole",
    "iam:DeletePolicy",
    "iam:DeleteRole",
    "iam:DetachRolePolicy",
    "iam:GetPolicy",
    "iam:GetRole",
    "iam:ListAttachedRolePolicies",
    "iam:ListPolicies",
    "iam:ListRoles",
    "iam:PassRole",
    "iam:PutRolePolicy",
    "iam:UpdateAssumeRolePolicy"
  ],
  "Effect": "Allow",
  "Resource": [
    "*"
  ]
} ,
```

If `Resource` is set to `"*"` as in the case above, then the user can pass any role.

For example, imagine that a user has permissions only to launch EC2 instances, but the roles that the user can pass to an EC2 instance have permissions to work with IAM and Amazon S3. In this case, the user might be able to launch the instance, log into it, get temporary security credentials and then perform IAM or S3 actions for which that the user is not authorised.

Recommendation:

The policies affected by this issue should be reviewed.

One potential mitigation is to specify the complete ARN of the role that the user should be allowed to pass. For example, the following policy sets this restriction:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:GetRole",
            "iam:PassRole"
        ],
        "Resource": "arn:aws:iam::<account-id>:role/<role-name>"
    }]
}
```

This helps prevent the user from running applications that have more permissions than the user has been granted - that is, from being able to obtain elevated privileges.

Affects:

IAM Policy
jenkins-role
jenkins-role1
servicerolepolicy
autoscalingclusterpolicy

References:

AWS Identity and Access Management (IAM)

https://aws.amazon.com/iam/

AWS Identity and Access Management Documentation

https://aws.amazon.com/documentation/iam/

AWS Documentation – Granting a User Permissions to Pass a Role to an AWS Service

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_passrole.html

AWS Documentation – Pass Role Permissions

https://docs.aws.amazon.com/iot/latest/developerguide/pass-role.html

AWS Documentation – Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html

AWS Security Blog – Granting Permission to Launch EC2 Instances with IAM Roles (PassRole Permission)

https://aws.amazon.com/blogs/security/granting-permission-to-launch-ec2-instances-with-iam-roles-passrole-permission

SR-128-3-9	*Default Security Group Hardening Not Implemented*	
Risk Rating	Medium	

Description:

A number of security groups were identified to be using the default rules, allowing all inbound traffic from instances assigned to the same security group, as well as all outbound traffic. The default security group is assigned to new instances created within a VPC if no custom security groups are assigned to it during configuration. These rules may be overly permissive, for instance allowing an attacker who has compromised one instance with the default security group assigned to use horizontal privilege escalation to compromise all other instances configured with the default security group.

As can be seen in the following screenshot from an NCC Group internal tool used during the assessment, the default security group had permissive rules defined:

Figure 14 – Rules present in default security group

The following VPCs were found to have default security groups configured with the default rules:

- region 1:
  - VPC 1
  - VPC 5
- region 2:
  - VPC a
  - VPC b

It is good security practice to harden the default security group by removing all rules. Any new instance created within a VPC will then be unable to communicate with other instances until custom security groups (conforming to security best practice) have been assigned.

Recommendation:

In order to improve system hardening, remove all rules from the default security group so that it restricts all traffic. Should an instance be created without custom security groups, it will inherit the default security group and be unable to communicate with other instances within the VPC until the required custom security groups are assigned.

Each VPC should be configured with custom security groups that conform to the practice of ‘least privilege’. These custom security groups should be assigned on a case-by-case basis to the instances that require them, allowing communication within instances and to the wider Internet under tightly controlled conditions.

Affects:

IP Address
VPC 1
VPC 5
VPC a
VPC b

References:

Amazon Virtual Private Cloud

https://aws.amazon.com/vpc/

Amazon Virtual Private Cloud Documentation

https://aws.amazon.com/documentation/vpc/

AWS Documentation – Security Groups for Your VPC

https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html

CIS Amazon Web Services Foundations

https://d0.awsstatic.com/whitepapers/compliance/AWS_CIS_Foundations_Benchmark.pdf

SR-128-3-10	*Single Availability Zone RDS Instance*	
Risk Rating	Low	

Description:

One Relational Database Service (RDS) instance was configured without the Multi Availability Zone option enabled. Without this, should an availability zone specific database failure occur, then Amazon RDS cannot automatically fail over to the standby availability zone so that database operations can resume quickly without administrative intervention.

The following database instance was configured with Multi-AZ disabled:

- components-db-stage

The following screenshot shows that Multi-AZ was disabled for the instance listed above:

REDACTED

Figure 15 – RDS Availability Zones

Recommendation:

Consideration should be given to modifying the configuration of the database instance to enable Multi-AZ. This is especially important if the applications using the database require high availability.

Affects:

RDS Instance
components-db-stage

References:

Amazon Relational Database Service (RDS)

https://aws.amazon.com/rds/

Amazon Relational Database Service Documentation

https://aws.amazon.com/documentation/rds/

AWS Documentation – High Availability (Multi-AZ)

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html

2.1.4 Phase 4 – Kubernetes Configuration

SR-128-4-1	*Kubernetes Pod Security Policy Overly Permissive*	
Risk Rating	Medium	

Description:

The cluster did make use of the Kubernetes Pod Security Policy feature, but was configured to allow any user to create privileged containers. As a result, any user with the ability to create new pods on the cluster would be able to gain privileged access to the underlying cluster nodes and any secrets contained within them.

Kubernetes provides this feature to restrict the privileges which can be granted by users to newly created pods. This can be used to prevent mounting of host file systems and other rights, which could allow unauthorised access to the underlying operating system running on the cluster nodes. The policies configured within the `project-staging` cluster did not force pods to run as non-root users, and allowed privileged containers to be created and run.

The output below provides evidence of the affected configuration option:

```
<REDACTED>
```

Recommendation:

Ensure that a Pod Security Policy is defined for all namespaces in the cluster. This policy should provide minimal rights restricting the ability of pods to gain escalated privileges within the cluster.

Affects:

Cluster Name
project-staging

References:

CIS Kubernetes Benchmark – Check 1.1.24
https://www.cisecurity.org/benchmark/Kubernetes/

Kubernetes Pod Security Policies

https://kubernetes.io/docs/concepts/policy/pod-security-policy/

Effective implantation of Pod Security Policies

https://octetz.com/posts/setting-up-psps

SR-128-4-2	*Kubernetes Unrestricted Cluster Network*	
Risk Rating	Medium	

Description:

There were no egress restrictions on traffic from pods in the cluster network. As a result, a compromised container would be able to attack Kubernetes' management interfaces.

By default, Kubernetes clusters provide an unrestricted flat network for all pods running on them. This can be restricted in recent versions by using Kubernetes Network Policies. These policies can be defined to restrict inbound and outbound access from the pods running on the cluster.

As this feature was not in place it was possible to contact services running in the cluster and the management services running on the nodes.

Recommendation:

Ensure that Kubernetes Network Policies are in place for all services running on the cluster. These policies should be constructed on a 'least privilege' basis to allow only required traffic.

Affects:

Cluster Name
project-staging

References:

CIS Kubernetes Benchmark Check 1.6.2
https://www.cisecurity.org/benchmark/Kubernetes/

Kubernetes Network Policy

https://kubernetes.io/docs/concepts/services-networking/network-policies/

SR-128-4-3	*Outdated Kubernetes in Use*	
Risk Rating	Medium	

Description:

The Kubernetes service was found to be out of date, leaving the clusters and their respective containers potentially open to both security and functional issues that have been patched in later versions.

Due to its very fast development rate, vulnerabilities that are found in Kubernetes often apply to numerous versions; however, patches are usually released soon after to remediate any current issues. Issues disclosed in previous versions have included privilege escalation and information disclosure issues, some of which could potentially allow a malicious user to take full control of a cluster.

Although no significant vulnerabilities were found in the observed versions of Kubernetes, it is strongly recommended that all clusters are kept up to date, in line with security best practice.

The code snippet below shows the current version of the Kubernetes clusters under review:

```
Server Version: version.Info{Major:"1", Minor:"16", GitVersion:"v1.16.2",
GitCommit:" xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx", GitTreeState:"clean",
BuildDate:"2019-10-20T09:56:31Z", Compiler:"gc", Platform:"linux/amd64"}
```

Recommendation:

Review the current patching policy for Kubernetes and ensure that all clusters are updated to the latest stable version regularly. It should also be noted that the current support life cycle length for versions of Kubernetes is nine months.

Affects:

Cluster
project-staging

3 Supplemental Data

The section below contains additional data that has been removed from the main body of the report for ease of readability.

3.1 Credentials in GitHub Repository

The files which were found to contain credentials are listed below:

```
<REDACTED>
```

3.2 List of the Identified Credentials

The following credentials were found in various GitHub files:

```
<REDACTED>
```

3.3 EBS Volumes Without Encryption

```
<REDACTED>
```

4 Appendices

4.1 Tool List

The following tools were used during the assessment:

Tools Used	Description
awscli	The AWS Command Line Interface https://aws.amazon.com/cli/
Burp Suite Pro	Intercepting proxy and web application scanner https://portswigger.net/
curl	Open source command line tool and library for transferring data with URL syntax https://curl.haxx.se
Kali Linux	Security-focussed Linux distribution (successor to Backtrack Linux) https://www.kali.org/
NCC Group Internal Tool	Security auditing tool for AWS environments
Nessus	Vulnerability scanner https://www.tenable.com/products/nessus
Nmap	Open source port scanner https://nmap.org/
udp-proto-scanner	UDP service discovery tool https://labs.portcullis.co.uk/application/udp-proto-scanner/

4.2 Assessment Team

The following members of staff were assigned to this assessment:

Name	Job Title	Comments
TSC Consultant	Senior Security Consultant	CREST Registered Tester (CRT), AWS Certified Security – Specialty, CREST Certified Simulated Attack Specialist (CCSAS)