CIS Benchmarks

CIS Amazon Web Services Three-tier Web Architecture Benchmark [imported] - ARCHIVE

v1.0.0 – 11-29-2016

Terms of Use

Please see the below link for our current terms of use:

https://www.cisecurity.org/cis-securesuite/cis-securesuite-membership-terms-of-use/

Table of Contents

Overview

This is the archive of the CIS Benchmark for Amazon Web Services Three-tier Web Architecture. CIS encourages you to migrate to a more recent, supported version of this technology.

This document provides prescriptive guidance for establishing a secure operational posture for a three-tier Web architecture deployed to the Amazon Web Services environment. Notionally, the three-tier Web architecture consists of a single Virtual Private Cloud (VPC) within a single AWS account. The recommendations made in the CIS AWS Foundations Benchmark should be followed prior to completing these recommendations. This benchmark covers the necessary AWS configurations to establish ongoing operations of a three-tier Web architecture.
Specific Amazon Web Services in scope for this document include:

- Elastic Compute Cloud (EC2) - API Version 2016-04-01
- Virtual Private Cloud (VPC) - API Version 2016-04-01
- Identity and Access Management (IAM) - API Version 2010-05-08
- AWS Config - API Version 2014-11-12
- CloudFront CDN - API Version 2016-01-13
- CloudWatch - API Version 2010-08-01
- Amazon Relational Database Service (RDS) - API Version 2014-10-31
- Simple Notification Service (SNS) - API Version 2010-03-31
- AWS Certificate Manager (ACM) - API Version 2015-12-08
- Key Management Service (KMS) - API Version 2014-11-01

While this Benchmark explicitly covers 3-tier architectures featuring Internet, Application and Database tiers (with a Content Distribution Network in the form of CloudFront, which could be considered a fourth tier), the tiers and the interactions between them are readily generalisable to larger "n-tier" architectures incorporating further tiers such as service proxy and management tiers. In particular, Security Group, routing, subnetting and VPC considerations and configurations can be re-used for further tiers and the segregation of communication between these tiers and others.
To obtain the latest version of this guide, please visit http://benchmarks.cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write us at feedback@cisecurity.org.

Intended Audience

This document is intended for system and application administrators, security specialists, auditors, help desk, platform deployment, and/or DevOps personnel who plan to develop, deploy, assess, or secure solutions in Amazon Web Services.

Consensus Guidance

This benchmark was created using a consensus review process comprised of subject matter experts. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

Each CIS benchmark undergoes two phases of consensus review. The first phase occurs during initial benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the benchmark. This discussion occurs until consensus has been reached on benchmark recommendations. The second phase begins after the benchmark has been published. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the benchmark. If you are interested in participating in the consensus process, please visit https://workbench.cisecurity.org/.

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
Monospace font	Used for inline code, commands, or examples. Text should be interpreted exactly as presented.
<italic font in brackets>	Italic texts set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to denote the title of a book, article, or other publication.
Note	Additional information or caveats

Scoring Information

A scoring status indicates whether compliance with the given recommendation impacts the assessed target's benchmark score. The following scoring statuses are used in this benchmark:

Scored

Failure to comply with "Scored" recommendations will decrease the final benchmark score. Compliance with "Scored" recommendations will increase the final benchmark score.

Not Scored

Failure to comply with "Not Scored" recommendations will not decrease the final benchmark score. Compliance with "Not Scored" recommendations will not increase the final benchmark score.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**

  Items in this profile intend to:

  - Be practical and prudent;
  - Provide a clear security benefit; and
  - Not inhibit the utility of the technology beyond acceptable means.

- **Level 2**

  This profile extends the "Level 1" profile. Items in this profile exhibit one or more of the following characteristics:

  - Are intended for environments or use cases where security is paramount
  - Acts as defense in depth measure
  - May negatively inhibit the utility or performance of the technology.

Acknowledgements

This benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Contributor
Rob Witoff , Coinbase
Dave Walker
Chris Launey
Gavin Fitzpatrick
Gregory Frascadore
John Robel
Justin Brown
Blake Frantz
Ionut Dragoi
Valentin Sever Radoi
Darwin Sanoy AWS Certified Solutions Architect Associate

Editor
Iben Rodriguez
Rael Daruszka , Center for Internet Security

Recommendations

1 Data Protection

This section provides recommendations for protecting data that transits the application stack.

1.1 Ensure a customer created Customer Master Key (CMK) is created for the Web-tier (Scored)

Profile Applicability:

- Level 2

Description:

AWS Key Management Service (KMS) by default provides service Customer Managed Keys (CMK). Customers also have the ability to create CMKs, which allows for configuration of key rotation and key policy which is applied to the customer created CMK.

You can use the key policy by itself to control who has access to the CMK and what actions each identity can perform. Controlling access this way specifies the full scope of access to the CMK in a single document (the key policy).

Customer created CMKs can be used for:

- AWS Service level encryption(e.g. EBS, RDS, S3)
- Key management for file/application level encryption

Rationale:

Ensures principle of least privilege on key ownership & usage

Audit:

Run the following command via the Amazon unified command line interface to determine if a web-tier KMS key exists:

```
aws kms list-aliases --query 'Aliases[?AliasName == `<web_tier_kms_alias>`]'
```

If such a key exists, an `AliasArn` property will be displayed in the output.

Remediation:

Using the Amazon unified command line interface:

- If there is no alias listed for Web tier, create new KMS key and note the "KeyId" element:

```
aws kms create-key
```

- Create an alias for the Web tier key using the above KeyId:

```
aws kms create-alias --alias-name <web_tier_kms_alias> <span class="pre">--
target-key-id <web_tier_kms_key>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/kms/list-aliases.html
2. http://docs.aws.amazon.com/cli/latest/reference/kms/create-key.html
3. http://docs.aws.amazon.com/cli/latest/reference/kms/create-alias.html

1.2 Ensure a customer created Customer Master Key (CMK) is created for the App-tier (Scored)

Profile Applicability:

- Level 2

Description:

AWS Key Management Service (KMS) by default provides service Customer Managed Keys (CMK). Customers also have the ability to create CMKs, which allows for configuration of key rotation and key policy which is applied to the customer created CMK.

You can use the key policy by itself to control who has access to the CMK and what actions each identity can perform. Controlling access this way specifies the full scope of access to the CMK in a single document (the key policy).

Customer created CMKs can be used for:

- AWS Service level encryption(e.g. EBS, RDS, S3)
- Key management for file/application level encryption

Rationale:

Ensures principle of least privilege on key ownership & usage

Audit:

Run the following command via the Amazon unified command line interface to determine if a web-tier KMS key exists:

```
aws kms list-aliases --query 'Aliases[?AliasName == `<app_tier_kms_alias>`]'
```

If such a key exists, an `AliasArn` property will be displayed in the output.

Remediation:

Using the Amazon unified command line interface:

- If there is no alias listed for App tier, create new KMS key and note the "KeyId" element:

```
aws kms create-key
```

- Create an alias for the App tier key using the above KeyId:

```
aws kms create-alias --alias-name <app_tier_kms_alias> --target-key-id
<app_tier_kms_key>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/kms/list-aliases.html
2. http://docs.aws.amazon.com/cli/latest/reference/kms/create-key.html
3. http://docs.aws.amazon.com/cli/latest/reference/kms/create-alias.html

1.3 Ensure a customer created Customer Master Key (CMK) is created for the Database-Tier (Scored)

Profile Applicability:

- Level 2

Description:

AWS Key Management Service (KMS) by default provides service Customer Managed Keys (CMK). Customers also have the ability to create CMKs, which allows for configuration of key rotation and key policy which is applied to the customer created CMK.

You can use the key policy by itself to control who has access to the CMK and what actions each identity can perform. Controlling access this way specifies the full scope of access to the CMK in a single document (the key policy).

Customer created CMKs can be used for:

- AWS Service level encryption(e.g. EBS, RDS, S3)
- Key management for file/application level encryption

Rationale:

Ensures principle of least privilege on key ownership & usage

Audit:

Run the following command via the Amazon unified command line interface to determine if a web-tier KMS key exists:

```
aws kms list-aliases --query 'Aliases[?AliasName == `<data_tier_kms_alias>`]'
```

If such a key exists, an `AliasArn` property will be displayed in the output.

Remediation:

Using the Amazon unified command line interface:

- If there is no alias listed for Data tier, create new KMS key and note the "KeyId" element:

```
aws kms create-key
```

- Create an alias for the Data tier key using the above KeyId:

```
aws kms create-alias --alias-name <data_tier_kms_alias> <span class="pre">--
target-key-id <data_tier_kms_key>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/kms/list-aliases.html
2. http://docs.aws.amazon.com/cli/latest/reference/kms/create-key.html
3. http://docs.aws.amazon.com/cli/latest/reference/kms/create-alias.html

1.4 Ensure Databases running on RDS have encryption at rest enabled (Scored)

Profile Applicability:

- Level 1

Description:

Amazon RDS instances and snapshots can be encrypted at rest by enabling the encryption option on the Amazon RDS DB instance. Data that is encrypted at rest includes the underlying storage for a DB instance, its automated backups, read replicas, and snapshots. It is recommended that encryption at rest be enabled.

Rationale:

Enabling encryption at rest will help ensure that the confidentiality of data stored in RDS, snapshots, and backups, is maintained.

Audit:

Using the Amazon unified CLI:

- List all current RDS instances and review the encryption status of the DB instance:

```
aws rds describe-db-instances --query 'DBInstances[*].{DBName:DBName,
EncryptionEnabled:StorageEncrypted, CMK:KmsKeyId}'
```

Remediation:

Using the Amazon unified CLI:

- Perform a snapshot of the DB instance:

```
aws rds create-db-snapshot --db-snapshot-identifier <db_snapshot> --db-
instance-identifier <your_db_instance>
```

- Confirm created snapshot is available (once snapshot process has completed):

```
aws rds describe-db-snapshots --query
'DBSnapshots[*].{DBSnapshotIdentifier:DBSnapshotIdentifier,
DBInstanceIdentifier:DBInstanceIdentifier, Snapshotstatus:Status}'
```

- List all KMS Customer Managed Keys:

```
aws kms list-aliases
```

- Copy to source RDS snapshot (from previous step) to a destination snapshot which will be encrypted:

```
aws rds copy-db-snapshot --source-db-snapshot-identifier <db_snapshot>  --
target-db-snapshot-identifier <encrypted_db_snapshot> --kms-key-id
<data_tier_kms_key>
```

- Restore a snapshot to the target DB instance(from previous step) with same values as original db instance with additional encrypted storage values:

```
aws rds restore-db-instance-from-db-snapshot --db-instance-identifier
<your_db_instance> --db-snapshot-identifier <encrypted_db_snapshot>
```

Impact:

This process will result in system down time from DB Snapshot (DB Backup) and restore to a new DB Instance. DB Snapshots are not instant - you will need to review the status of the snapshot until it has completed.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-instances.html
2. http://docs.aws.amazon.com/cli/latest/reference/rds/create-db-snapshot.html
3. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-snapshots.html
4. http://docs.aws.amazon.com/cli/latest/reference/rds/create-db-instance.html
5. http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CopySnapshot.html
6. http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html
7. http://docs.aws.amazon.com/cli/latest/reference/rds/copy-db-snapshot.html
8. http://docs.aws.amazon.com/cli/latest/reference/kms/list-aliases.html

1.5 Ensure all EBS volumes for Web-Tier are encrypted (Scored)

Profile Applicability:

- Level 1

Description:

Elastic Block Storage (EBS) volumes can be encrypted using AWS Key Management Service (KMS). In this configuration, encryption and decryption are handled transparently and require no additional action from the user, an Amazon EC2 instance, or application. When an encrypted Amazon EBS volume is attached to a supported Amazon EC2 instance type, the data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. The encryption occurs on the servers that host Amazon EC2 instances. Additionally, snapshots of encrypted volumes are automatically encrypted, and volumes that are created from encrypted snapshots are also automatically encrypted. It is recommended that all EBS volumes for the web tier be encrypted.

Rationale:

Enabling encryption on EBS volumes will help ensure the confidentiality of data stored on those volumes.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<web\_tier\_tag>`:<web_tier_tag_value> with your own tag and value for the Web tier)

- Note the EBS volume id's, Instance id's, Availability Zones, and check if `Encrypted` element is `True` or `False`

```
aws ec2 describe-volumes --filters Name=tag:<web_tier_tag>,
Values=<web_tier_tag_value> --query 'Volumes[*].{VolumeId:VolumeId,
Encrypted:Encrypted, AvailabilityZone:AvailabilityZone,
InstanceId:Attachments[*].InstanceId}' --output table
```

Remediation:

Using the Amazon unified command line interface:

- Note all the volume id's of unencrypted EBS volumes and create a snapshot for each of them:

```
aws ec2 create-snapshot --volume-id <unencrypted_ebs_volume> --description
"Snapshot for encryption operation"
```

- Note the `SnapshotId`element from the output of step 1 and copy the snapshot to an encrypted snapshot using the KMS key created for the Web-tier:

```
aws ec2 copy-snapshot --source-region <application_region> --source-snapshot-
id <unencrypted_ebs_snapshot> --description "Encrypted snapshot." --encrypted
--kms-key-id <web_tier_kms_key>
```

- Note the `SnapshotId` element from the output of step 2 and create a new EBS volume from the encrypted snapshot in the same Availability Zone as the unencrypted volume:

```
aws ec2 create-volume --availability-zone <application_az> --snapshot-id
<encrypted_ebs_snapshot>
```

- Tag the newly created EBS volume using the Volume Id from the previous step *<encrypted_ebs_volume>:*

```
aws ec2 create-tags --resources <encrypted_ebs_volume> --tags
Key=<web_tier_tag>,Value=<web_tier_tag_value>
```

- Delete unencrypted EBS volume:

```
aws ec2 delete-volume --volume-id <unencrypted_ebs_volume>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-volumes.html
2. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html#EBSEncryption_considerations
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-snapshot.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/copy-snapshot.html
5. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-volume.html
6. http://docs.aws.amazon.com/cli/latest/reference/ec2/delete-volume.html

1.6 Ensure all EBS volumes for App-Tier are encrypted (Scored)

Profile Applicability:

- Level 1

Description:

Elastic Block Storage (EBS) volumes can be encrypted using AWS Key Management Service (KMS). In this configuration, encryption and decryption are handled transparently and require no additional action from the user, an Amazon EC2 instance, or application. When an encrypted Amazon EBS volume is attached to a supported Amazon EC2 instance type, the data stored at rest on the volume, disk I/O, and snapshots created from the volume are all encrypted. The encryption occurs on the servers that host Amazon EC2 instances. Additionally, snapshots of encrypted volumes are automatically encrypted, and volumes that are created from encrypted snapshots are also automatically encrypted. It is recommended that all EBS volumes for the app tier be encrypted.

Rationale:

Enabling encryption on EBS volumes will help ensure the confidentiality of data stored on those volumes.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<app\_tier\_tag>:<app_tier_tag_value>` with your own tag and value for the App tier)

- Note the EBS volume id's, Instance id's, Availability Zones, and check if `Encrypted` element is `True` or `False`

```
aws ec2 describe-volumes --filters
Name=tag:<app_tier_tag>,Values=<app_tier_tag_value> --query
'Volumes[*].{VolumeId:VolumeId, Encrypted:Encrypted,
AvailabilityZone:AvailabilityZone, InstanceId:Attachments[*].InstanceId}' --
output table
```

Remediation:

Using the Amazon unified command line interface:

- Note all the volume id's of unencrypted EBS volumes and create a snapshot for each of them:

```
aws ec2 create-snapshot --volume-id <unencrypted_ebs_volume> --description
"Snapshot for encryption operation"
```

- Note the `SnapshotId`element from the output of step 1 and copy the snapshot to an encrypted snapshot using the KMS key created for the Web-tier:

```
aws ec2 copy-snapshot --source-region <application_region> --source-snapshot-
id <unencrypted_ebs_snapshot> --description "Encrypted snapshot." --encrypted
--kms-key-id <app_tier_kms_key>
```

- Note the `SnapshotId` element from the output of step 2 and create a new EBS volume from the encrypted snapshot in the same Availability Zone as the unencrypted volume:

```
aws ec2 create-volume --availability-zone <application_az> --snapshot-id
<encrypted_ebs_snapshot>
```

- Tag the newly created EBS volume using the Volume Id from the previous step *<encrypted_ebs_volume>:*

```
aws ec2 create-tags --resources <encrypted_ebs_volume> --tags
Key=<app_tier_tag>,<app_tier_tag_value>
```

- Delete unencrypted EBS volume:

```
aws ec2 delete-volume --volume-id <unencrypted_ebs_volume>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-volumes.html
2. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html#EBSEncryption_considerations
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-snapshot.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/copy-snapshot.html
5. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-volume.html
6. http://docs.aws.amazon.com/cli/latest/reference/ec2/delete-volume.html

1.7 Ensure all Customer owned Amazon Machine Images for Web Tier are not shared publicly (Scored)

Profile Applicability:

- Level 1

Description:

Amazon Machine Images (AMI) are an exact duplicate of the instance they were created from and will allow anyone with access to create a complete replica of the original instance. The original instance may contain intellectual property, proprietary applications, and configuration information that can be used to exploit or compromise any running instance in the web tier.

Rationale:

Allowing public access to the Web Tier AMI may aid an adversary in identifying weaknesses in the application use or configuration.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<web\_tier\_tag>:<web_tier_tag_value>` with your own tag and value for the Web tier)

- Note the image id's, AMI name, and check if `Public` element is `True` or `False`:

```
aws ec2 describe-images --owners self  --filters
Name=tag:<web_tier_tag>,Values=<web_tier_tag_value> --query
'Images[*].{Name:Name, ImageId:ImageId, Public:Public}' --output table
```

Remediation:

Using the Amazon unified command line interface:

- For each AMI that is public remove group `ALL` from the launch permissions:

```
aws ec2 modify-image-attribute --image-id <public_image_id> --launch-
permission "{\"Remove\":[{\"Group\":\"all\"}]}"
```

Default Value:

The prescribed value is the default value.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-images.html
2. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/sharingamis-intro.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/modify-image-attribute.html

1.8 Ensure all Customer owned Amazon Machine Images for Application Tier are not shared publicly (Scored)

Profile Applicability:

- Level 1

Description:

Amazon Machine Images are an exact duplicate of the instance they were created from and will allow anyone with access to create a complete replica of the original instance. The original instance may contain intellectual property, proprietary applications, and configuration information that can be used to exploit or compromise any running instance in the web tier.

Rationale:

Allowing public access to the Application Tier AMI may aid an adversary in identifying weaknesses in the application use or configuration.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<app\_tier\_tag>:<app_tier_tag_value>` with your own tag and value for the App-tier)

- note the image id's, AMI name, and check if `Public` element is `True` or `False`:

```
aws ec2 describe-images --owners self  --filters
Name=tag:<app_tier_tag>,Values=<app_tier_tag_value> --query
'Images[*].{Name:Name, ImageId:ImageId, Public:Public}' --output table
```

Remediation:

Using the Amazon unified command line interface:

- For each AMI that is public remove group `ALL` from the launch permissions:

```
aws ec2 modify-image-attribute --image-id <public_image_id> --launch-
permission "{\"Remove\":[{\"Group\":\"all\"}]}"
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-images.html
2. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/sharingamis-intro.html

3. http://docs.aws.amazon.com/cli/latest/reference/ec2/modify-image-attribute.html

1.9 Ensure Web Tier ELB have SSL/TLS Certificate attached (Scored)

Profile Applicability:

- Level 1

Description:

When you use HTTPS for your front-end listener, you must deploy an SSL/TLS certificate on your load balancer. The load balancer uses the certificate to terminate the connection and then decrypt requests from clients before sending them to the back-end instances.

The TLS protocol uses an X.509 certificate (SSL/TLS server certificate) to authenticate both the client and the back-end application. An X.509 certificate is a digital form of identification issued by a trusted certificate authority (CA) and contains identification information, a validity period, a public key, a serial number, and the digital signature of the issuer.

You can create a certificate using a Third Party Certificate Authority or AWS Certificate Manager.

- Note: an SSL certificate configured on the ELB is not mandatory if you are terminating SSL connections directly on the Web Tier EC2 instances, and using a TCP listener on the ELB (TCP pass-through)

Rationale:

All the application traffic between the clients and the Web Tier ELB nodes should be encrypted using a SSL/TLS certificate.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<web\_tier\_elb>` with your Web-tier ELB name)

- Note that if the `ListenerDescriptions` element is empty, the ELB does not have a listener configured with a SSL/TLS certificate, or a TCP listener (TCP pass-through):

```
aws elb describe-load-balancers --load-balancer-names <web_tier_elb> --query
"LoadBalancerDescriptions[*].{LoadBalancerName:LoadBalancerName,
DNSName:DNSName, Scheme:Scheme,
ListenerDescriptions:ListenerDescriptions[?Listener.SSLCertificateId != null
||  Listener.Protocol == 'TCP']}" --output table
```

Remediation:

Using the Amazon unified command line interface:

- Adding a HTTPS listener configured with a SSL/TLS certificate (the listener forwards traffic to the backend instances on port 80, but this can be modified by editing `InstancePort=80`):

```
aws elb create-load-balancer-listeners --load-balancer-name <web_tier_elb> --
listeners
Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,
SSLCertificateId=<ssl_certificate_arn>
```

References:

1. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-add-or-delete-listeners.html#add-listener-cli
2. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html

1.10 Ensure Web Tier ELB have the latest SSL Security Policies configured (Scored)

Profile Applicability:

- Level 1

Description:

Elastic Load Balancing uses an Secure Socket Layer (SSL) negotiation configuration, known as a security policy, to negotiate SSL/TLS connections between a client and the load balancer. A security policy is a combination of SSL/TLS protocols, ciphers, and the Server Order Preference option.

Elastic Load Balancing supports configuring your load balancer to use either predefined or custom security policies.

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptographic protocols that are used to encrypt confidential data over insecure networks such as the Internet. The TLS protocol is a newer version of the SSL protocol. In the Elastic Load Balancing documentation, we refer to both SSL and TLS protocols as the SSL protocol.

- Note: an SSL certificate configured on the ELB and an SSL Security Policy is not mandatory if you are terminating SSL connections directly on the Web Tier EC2 instances, and using a TCP listener on the ELB (TCP pass-through)

Rationale:

Making sure the latest ELB SSL Security Policy is used will ensure the SSL/TLS connection will be negotiated using only the appropriate cryptographic protocols deemed safe with no proven vulnerabilities.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<web\_tier\_elb>` with your Web-tier ELB name)

- Find all the SSL security policies associated with your load balancer listener:

```
aws elb describe-load-balancer-policies --load-balancer-name <web_tier_elb> --query
'PolicyDescriptions[?PolicyTypeName==`SSLNegotiationPolicyType`].{PolicyName:PolicyName,ReferenceSecurityPolicy:PolicyAttributeDescriptions[0].AttributeValue}' --output table
```

- Find which of the above policies is currently active, and check on AWS documentation if it is the latest (note that for the TCP listeners the `PolicyNames` element will be empty, but the TCP listener is still compliant when using SSL certificates on the back-end EC2 instances):

```
aws elb describe-load-balancers --load-balancer-name <web_tier_elb> --query
"LoadBalancerDescriptions[*].{CompliantListeners:ListenerDescriptions[?Listen
er.SSLCertificateId != null || Listener.Protocol == 'TCP']}" --output table
```

Remediation:

Using the Amazon unified command line interface:
(Note that you should replace `<web\_tier\_elb>` with your Web-tier ELB name, and `<latest\_ssl\_policy>_` with the proper policy name)

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name
<web_tier_elb> --load-balancer-port 443 --policy-names  <latest_ssl_policy>
```

References:

1. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-security-policy-options.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/set-load-balancer-policies-of-listener.html
3. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancer-policies.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
5. https://aws.amazon.com/premiumsupport/knowledge-center/elb-listener-policy-cli/
6. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-security-policy-options.html

1.11 Ensure Web Tier ELB is using HTTPS listener (Scored)

Profile Applicability:

- Level 2

Description:

A load balancer takes requests from clients and distributes them across the EC2 instances that are registered with the load balancer (also known as back-end instances).

A listener is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections

- Note: an HTTPS listener configured on the ELB is not mandatory if you are terminating SSL connections directly on the Web Tier EC2 instances, and using a TCP listener on the ELB (TCP pass-through)

Rationale:

Using an HTTPS Elastic Load Balancer listener will make sure the application traffic between the client and the Web Tier ELB is encrypted over the SSL\TLS channel.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<web\_tier\_elb>` with your Web-tier ELB name)

- Check if the Web Tier ELB is using an HTTPS or TCP listener. Note if the `ListenerDescription` field is missing or not:

```
aws elb describe-load-balancers --load-balancer-names  <web_tier_elb> --query
"LoadBalancerDescriptions[*].{LoadBalancerName:LoadBalancerName,
DNSName:DNSName, Scheme:Scheme,
ListenerDescriptions:ListenerDescriptions[?Listener.Protocol == 'HTTPS' ||
Listener.Protocol = 'TCP']}" --output table
```

Remediation:

Using the Amazon unified command line interface:

- If the ListenerDescription field is missing, add a new HTTPS listener configured with a SSL\TLS certificate (the listener forwards traffic to the backend instances on port 80, but this can be modified by editing InstancePort=80):

```
aws elb create-load-balancer-listeners --load-balancer-name <web_tier_elb> --
listeners
Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,
SSLCertificateId=<ssl_certificate_arn>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/create-load-balancer-listeners.html
3. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-listener-config.html

1.12 Ensure App Tier ELB have SSL\TLS Certificate attached (Scored)

Profile Applicability:

- Level 2

Description:

When you use HTTPS for your front-end listener, you must deploy an SSL/TLS certificate on your load balancer. The load balancer uses the certificate to terminate the connection and then decrypt requests from clients before sending them to the back-end instances.

The SSL\TLS protocol uses an X.509 certificate (SSL\TLS server certificate) to authenticate both the client and the back-end application. An X.509 certificate is a digital form of identification issued by a trusted certificate authority (CA) and contains identification information, a validity period, a public key, a serial number, and the digital signature of the issuer.

You can create a certificate using a Third Party Certificate Authority, AWS Certificate Manager or a self signed certificate like OpenSSL.

- Note: an SSL certificate configured on the ELB is not mandatory if you are terminating SSL connections directly on the App Tier EC2 instances, and using a TCP listener on the ELB (TCP pass-through)

Rationale:

All the application traffic between the Web Tier instances and the App Tier ELB nodes should be encrypted using an SSL\TLS certificate.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<app\_tier\_elb>` with your App-tier ELB name)

- Note that if the `ListenerDescriptions` is empty, the ELB does not have a listener configured with a SSL\TLS certificate, or a TCP listener (TCP pass-through):

```
aws elb describe-load-balancers --load-balancer-names <app_tier_elb> --query
"LoadBalancerDescriptions[*].{LoadBalancerName:LoadBalancerName,
DNSName:DNSName, Scheme:Scheme,
ListenerDescriptions:ListenerDescriptions[?Listener.SSLCertificateId != null
|| Listener.Protocol == 'TCP']}" --output table
```

Remediation:

Using the Amazon unified command line interface:

- Adding a HTTPS listener configured with a SSL\TLS certificate (the listener forwards traffic to the backend instances on port 80, but this can be modified by editing InstancePort=80):

```
aws elb create-load-balancer-listeners --load-balancer-name <app_tier_elb> --
listeners
Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,
SSLCertificateId=<ssl_certificate_arn>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
2. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-add-or-delete-listeners.html#add-listener-cli

1.13 Ensure App Tier ELB have the latest SSL Security Policies configured (Scored)

Profile Applicability:

- Level 2

Description:

Elastic Load Balancing uses a Secure Socket Layer (SSL) negotiation configuration, known as a security policy, to negotiate SSL/TLS connections between a client and the load balancer. A security policy is a combination of SSL/TLS protocols, ciphers, and the Server Order Preference option.

Elastic Load Balancing supports configuring your load balancer to use either predefined or custom security policies.

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are cryptographic protocols that are used to encrypt confidential data over insecure networks such as the Internet. The TLS protocol is a newer version of the SSL protocol. In the Elastic Load Balancing documentation, we refer to both SSL and TLS protocols as the SSL protocol.

- Note: an SSL certificate configured on the ELB and an SSL Security Policy is not mandatory if you are terminating SSL connections directly on the App Tier EC2 instances, and using a TCP listener on the ELB (TCP pass-through)

Rationale:

Making sure the latest ELB SSL Security Policy is used will ensure the SSL/TLS connection will be negotiated using only the appropriate cryptographic protocols deemed safe with no proven vulnerabilities.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<app\_tier\_elb>` with your App-tier ELB name)

- Find all the SSL security policies associated with your load balancer listener:

```
aws elb describe-load-balancer-policies --load-balancer-name <app_tier_elb> --query
'PolicyDescriptions[?PolicyTypeName==`SSLNegotiationPolicyType`].{PolicyName:PolicyName,ReferenceSecurityPolicy:PolicyAttributeDescriptions[0].AttributeValue}' --output table
```

- Find which of the above policies is currently active, and check on AWS documentation if it is the latest (note that for the TCP listeners the `PolicyNames` element will be empty, but the TCP listener is still compliant when using SSL certificates on the back-end EC2 instances)::

```
aws elb describe-load-balancers --load-balancer-name <app_tier_elb> --query
"LoadBalancerDescriptions[*].{CompliantListeners:ListenerDescriptions[?Listen
er.SSLCertificateId != null || Listener.Protocol == 'TCP']}" --output table
```

Remediation:

Using the Amazon unified command line interface:
(Note that you should replace `<app\_tier\_elb>` with your App-tier ELB name, and *`<latest\_ssl\_policy>`* with the proper policy name)

```
aws elb set-load-balancer-policies-of-listener --load-balancer-name
<app_tier_elb> --load-balancer-port 443 --policy-names  <latest_ssl_policy>
```

References:

1. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-security-policy-options.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/set-load-balancer-policies-of-listener.html
3. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancer-policies.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
5. https://aws.amazon.com/premiumsupport/knowledge-center/elb-listener-policy-cli/
6. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-security-policy-options.html

1.14 Ensure App Tier ELB is using HTTPS listener (Scored)

Profile Applicability:

- Level 2

Description:

A load balancer takes requests from clients and distributes them across the EC2 instances that are registered with the load balancer (also known as back-end instances).

A listener is a process that checks for connection requests. It is configured with a protocol and a port for front-end (client to load balancer) connections.

- Note: an HTTPS listener configured on the ELB is not mandatory if you are terminating SSL connections directly on the App Tier EC2 instances, and using a TCP listener on the ELB (TCP pass-through)

Rationale:

Using an HTTPS Elastic Load Balancer listener will make sure the application traffic between the client and the App Tier ELB is encrypted over the SSL\TLS channel.

Audit:

Using the Amazon unified command line interface:
(Note that you should replace `<app\_tier\_elb>` with your App tier ELB name)

- Check if the App Tier ELB is using an HTTPS or TCP listener. Note if the `ListenerDescription` field is missing or not:

```
aws elb describe-load-balancers --load-balancer-names  <app_tier_elb> --query
"LoadBalancerDescriptions[*].{LoadBalancerName:LoadBalancerName,
DNSName:DNSName, Scheme:Scheme,
ListenerDescriptions:ListenerDescriptions[?Listener.Protocol == 'HTTPS' ||
Listener.Protocol == 'TCP']}" --output table
```

Remediation:

Using the Amazon unified command line interface:

- If the `ListenerDescription` field is missing, add a new HTTPS listener configured with a SSL\TLS certificate (the listener forwards traffic to the backend instances on port 80, but this can be modified by editing `InstancePort=80`):

```
aws elb create-load-balancer-listeners --load-balancer-name <app_tier_elb> --
listeners
Protocol=HTTPS,LoadBalancerPort=443,InstanceProtocol=HTTP,InstancePort=80,
SSLCertificateId=<ssl_certificate_arn>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/create-load-balancer-listeners.html
3. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-listener-config.html

1.15 Ensure all Public Web Tier SSL\TLS certificates are >30 days from Expiration (Scored)

Profile Applicability:

- Level 2

Description:

Public SSL\TLS certificates that are used for AWS resources such as the ELB or CloudFront should always be renewed prior to expiration both as a security best practice and to ensure the reputation of the web application is not impacted by an expired certificate.

Rationale:

SSL\TLS certificates that are public should be renewed prior to expiration.

Audit:

Using the Amazon unified command line interface:

- List all SSL\TLS certificates stored in IAM and check the Expiration field

```
aws iam list-server-certificates
```

For Amazon Certificate Manager users:

- List ACM certificates and note the `CertificateArn` value of the certificates used by the app:

```
aws acm list-certificates
```

- Get the details of the desired certificate and check the `ExpirationDate` value (the certificate expiration time as an Epoch timestamp):

```
aws acm describe-certificate --certificate-arn <ssl_certificate_arn> --
query "Certificate.{ExpirationDate:NotAfter, Status:Status,
SubjectAlternativeNames:SubjectAlternativeNames,
DomainName:DomainName}"
```

Remediation:

Using the Amazon unified command line interface:

- Request a certificate renewal from your CA, and upload the new certificate in IAM:

```
aws iam upload-server-certificate --server-certificate-name
<ssl_certificate_name> --certificate-body
file://public_key_cert_file.pem --private-key file://my_private_key.pem
--certificate-chain file://my_certificate_chain_file.pem
```

- For Amazon Certificate Manager users the renewal is managed by ACM service

References:

1. http://docs.aws.amazon.com/cli/latest/reference/iam/list-server-certificates.html
2. http://docs.aws.amazon.com/cli/latest/reference/iam/upload-server-certificate.html
3. http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_server-certs.html
4. http://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_server-certs_manage.html
5. https://docs.aws.amazon.com/cli/latest/reference/acm/describe-certificate.html
6. https://docs.aws.amazon.com/cli/latest/reference/acm/list-certificates.html

1.16 Ensure all S3 buckets have policy to require server-side and in transit encryption for all objects stored in bucket. (Scored)

Profile Applicability:

- Level 1

Description:

Data in transit is data being accessed over the network, and therefore could be intercepted by someone else on the network or with access to the physical media the network uses. On an ethernet network, that could be someone with the ability to tap a cable, configure a switch to mirror traffic, or fool your client or a router into directing traffic to them before it moves on to the final destination.

Server-side encryption (SSE) is about data encryption at rest—that is, Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers and decrypts it for you when you access it.

Amazon S3 offers 3 options of encrypting data at rest, depending on how you choose to manage the encryption keys:

- Use SSE with Amazon S3-Managed Keys
- Use SSE with AWS KMS-Managed Keys
- Use SSE with Customer-Provided Keys

At the time of object creation—that is, when you are uploading a new object or making a copy of an existing object—you can specify if you want Amazon S3 to encrypt your data by adding the "x-amz-server-side-encryptionheader" to the request. Set the value of the header to the encryption algorithm AES256 that Amazon S3 supports. Amazon S3 confirms that your object is stored using server-side encryption by returning the response header "x-amz-server-side-encryption".

No matter which of the three options you choose, you can create and attach a S3 bucket policy, that will deny any object creation S3 API (PUT Object, PUT Object - Copy, POST Object, Initiate Multipart Upload), if the request does not include the "x-amz-server-side-encryption" header requesting server-side encryption, and if the request was not done using SSL\TLS.

Rationale:

When it comes to data at rest, if kept unencrypted, there are a few threats that one can think of, especially when the data is stored in the cloud:

- the threat that attackers are able to compromise Amazon S3 and gain access to the data that is stored in the Amazon S3 buckets.
- the "insider threat" where a malicious or rogue administrator steals a physical disk drive or server that contains data a customer has in the Amazon S3 buckets.
- the threat that a government uses a subpoena or warrant to get access to a customer's data in Amazon S3 without their knowledge.

If data in transit is kept unencrypted:

- Malicious users may intercept or monitor plaintext data transmitting across unencrypted network and gain unauthorized access to that jeopardize the confidentiality of the sensitive data.

In all of these scenarios, encrypting data at rest and in transit, and properly managing the encryption keys can help mitigate the risk of unauthorized access to that data.

Audit:

Using the Amazon unified command line interface:

- List all the S3 buckets from the AWS account:

```
aws s3api list-buckets --output table
```

- For each S3 bucket, list the bucket policy and verify that a policy exists, the policy contains a statement to deny `PutObject` calls that do not require server-side encryption, and the policy denies requests that do not occur over a secure transport:

```
aws s3api get-bucket-policy --bucket <s3_bucket_name> --query 'Policy'
...
    {
      "Sid": "DenyUnEncryptedObjectUploads",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::/*",
      "Condition": {
          "Null": { "s3:x-amz-server-side-encryption": "true",
          "Bool": { "aws:SecureTransport": "false" }
        }
      }
```

Remediation:

Perform the following to ensure all objects placed in S3 are encrypted in transit and at rest:

- Create a new file, add the following to it, and save it as `policy.json`:

```
 {
   "Version":"2012-10-17",
   "Id":"PutObjPolicy",
   "Statement":{
      "Sid":"DenyUnEncryptedObjectUploads",
      "Effect":"Deny",
      "Principal":"*",
      "Action":"s3:PutObject",
      "Resource":"arn:aws:s3:::<s3_bucket_name>/*",
      "Condition":{
         "Null":{ "s3:x-amz-server-side-encryption":"true" },
         "Bool": { "aws:SecureTransport": "false" }
      }
   }
}
```

- Attach the above bucket policy to each S3 bucket:

```
aws s3api put-bucket-policy --bucket <s3_bucket_name> --policy
file://policy.json
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/s3api/list-buckets.html
2. http://docs.aws.amazon.com/cli/latest/reference/s3api/get-bucket-policy.html
3. http://docs.aws.amazon.com/cli/latest/reference/s3api/put-bucket-policy.html
4. http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingServerSideEncryption.html

1.17 Ensure CloudFront to Origin connection is configured using TLS1.1+ as the SSL\TLS protocol (Scored)

Profile Applicability:

- Level 2

Description:

CloudFront can connect to your origin using only HTTP, only HTTPS, or to connect by matching the protocol used by the viewer. Our recommendation is to use HTTPS only. In this case you can choose which SSL\TLS protocols CloudFront is allowed to use when establishing an HTTPS connection to your origin. Ensure that you are using only TLS1.1+ as the SSL\TLS protocol.

Rationale:

It used to be believed that TLS v1 was marginally more secure than SSL v3.0, its predecessor. However, developments, such as the POODLE vulnerability have shown that SSL v3.0 is now insecure.

Subsequent versions of TLS — v1.1 and v1.2 are significantly more secure and fix many vulnerabilities present in SSL v3.0 and TLS v1. For example, the BEAST attack that can completely break web sites running on older SSL v3.0 and TLS v1 protocols. The newer TLS versions, if properly configured, prevent the BEAST and other attack vectors and provide many stronger ciphers and encryption methods.

Audit:

Using the Amazon unified command line interface:

- Check the "OriginSslProtocols" used by application Cloudfront distributions (only TLSv1.1 and TLSv1.2 should be present):

```
aws cloudfront list-distributions --query "DistributionList.Items[*].{Id:Id,
OriginSslProtocols:Origins.Items[*].CustomOriginConfig.OriginSslProtocols.Ite
ms[*]}"
```

Remediation:

Using the Amazon unified command line interface:

- For configuring Origin SSL protocols first save locally the current distribution config:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "DistributionConfig" > /tmp/cf-distribution.json
```

- Edit and replace "OriginSslProtocols" element in /tmp/cf-distribution.json with the below section:

```
"OriginSslProtocols": {
  "Items": [
    "TLSv1.1",
    "TLSv1.2"
  ],
  "Quantity": 2
},
```

- Retrieve the current ETag of your CloudFront distribution:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "ETag"
```

- Update the CloudFront distribution using the edited config and the above Etag:

```
aws cloudfront update-distribution --id <application_cfn_distribution_id> --
distribution-config file:///tmp/cf-distribution.json --if-match
<application_cfn_distribution_etag>
```

References:

1. https://aws.amazon.com/about-aws/whats-new/2016/01/amazon-cloudfront-adds-new-origin-security-features/
2. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/get-distribution.html
3. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/update-distribution.html

2 Identity and Access Management

This section provides recommendations on managing identities and access to resources.

2.1 Ensure IAM Policy for EC2 IAM Roles for Web tier is configured (Scored)

Profile Applicability:

- Level 1

Description:

By default, IAM users, groups, and roles have no access to AWS resources.

IAM policies are the means by which privileges are granted to users, groups, or roles defined with AWS Identity Access Management.

An IAM policy is a document that formally states one or more permissions using the following structure:

- Actions: what actions are allowed (each AWS service has its own set of actions)
- Resources: which resources will be affected by the action
- Effect: what effect will be when the subject (user/group/roles) requests access

Policies are documents that are created using JSON. A policy consists of one or more statements, each of which describes one set of permissions.

Rationale:

Ensure IAM policy defines a minimum level of access to AWS services : S3, Cloudwatch, KMS

Audit:

Using the Amazon unified command line interface:

- Check and note the IAM role name used by the Web tier instance profile:

```
aws iam get-instance-profile --instance-profile-name
<web_tier_instance_profile> --query "InstanceProfile.Roles[*].RoleName"
```

- Check managed policies attached to the IAM role, and note the policies ARN:

```
aws iam list-attached-role-policies --role-name <web_tier_iam_role>
```

- Check and note the version of the IAM policies attached to the IAM role:

```
aws iam get-policy --policy-arn <iam_policy_arn> --query
"Policy.DefaultVersionId"
```

- Check the document policy:

```
aws iam get-policy-version --policy-arn <iam_policy_arn> --version-id
<iam_policy_version>
```

Remediation:

Using the Amazon unified command line interface:

- If doesn't exist, create an instance profile for Web tier instances:

```
aws iam create-instance-profile --instance-profile-name
<web_tier_instance_profile>
```

- If doesn't exist, create an IAM role for the instance profile:
  - Create a trust relationship policy document and save it locally as /tmp/TrustPolicy.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

  - Create the IAM role using the above trust policy:

```
aws iam create-role --role-name <web_tier_iam_role> --assume-role-
policy-document file:///tmp/TrustPolicy.json
```

- Add the IAM role created to the Instance profile:

```
aws iam add-role-to-instance-profile --role-name <web_tier_iam_role> --
instance-profile-name <web_tier_instance_profile>
```

- If doesn't exist, create an IAM managed policy for Web tier instances, and note the policy ARN:

```
aws iam create-policy --policy-name <iam_policy_name> --policy-document
file://policy
```

- Attach the IAM policy created to the Web tier IAM role:

```
aws iam attach-role-policy --policy-arn <iam_policy_arn> --role-name
<web_tier_iam_role>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/iam/get-instance-profile.html
2. http://docs.aws.amazon.com/cli/latest/reference/iam/list-attached-role-policies.html
3. http://docs.aws.amazon.com/cli/latest/reference/iam/get-policy.html
4. http://docs.aws.amazon.com/cli/latest/reference/iam/get-policy-version.html
5. http://docs.aws.amazon.com/cli/latest/reference/iam/create-instance-profile.html
6. http://docs.aws.amazon.com/cli/latest/reference/iam/create-role.html
7. http://docs.aws.amazon.com/cli/latest/reference/iam/add-role-to-instance-profile.html
8. http://docs.aws.amazon.com/cli/latest/reference/iam/create-policy.html
9. http://docs.aws.amazon.com/cli/latest/reference/iam/attach-role-policy.html

2.2 Ensure IAM Policy for EC2 IAM Roles for App tier is configured (Scored)

Profile Applicability:

- Level 1

Description:

By default, IAM users, groups, and roles have no access to AWS resources.

IAM policies are the means by which privileges are granted to users, groups, or roles defined with AWS Identity Access Management.

An IAM policy is a document that formally states one or more permissions using the following structure:

- Actions: what actions are allowed (each AWS service has its own set of actions)
- Resources: which resources will be affected by the action
- Effect: what effect will be when the subject (user/group/roles) requests access

Policies are documents that are created using JSON. A policy consists of one or more statements, each of which describes one set of permissions.

Rationale:

Ensure IAM policy defines a minimum level of access to AWS services : S3, Cloudwatch, KMS

Audit:

Using the Amazon unified command line interface:

- Check and note the IAM role name used by the App tier instance profile:

```
aws iam get-instance-profile --instance-profile-name
<app_tier_instance_profile> --query "InstanceProfile.Roles[*].RoleName"
```

- Check managed policies attached to the IAM role, and note the policies ARN:

```
aws iam list-attached-role-policies --role-name <app_tier_iam_role>
```

- Check and note the version of the IAM policies attached to the IAM role:

```
aws iam get-policy --policy-arn <iam_policy_arn> --query
"Policy.DefaultVersionId"
```

- Check the document policy:

```
aws iam get-policy-version --policy-arn <iam_policy_arn> --version-id
<iam_policy_version>
```

Remediation:

Using the Amazon unified command line interface:

- If doesn't exist, create an instance profile for App tier instances:

```
aws iam create-instance-profile --instance-profile-name
<app_tier_instance_profile>
```

- If doesn't exist, create an IAM role for the instance profile:
    - Create a trust relationship policy document and save it locally as /tmp/TrustPolicy.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

    - Create the IAM role using the above trust policy:

```
aws iam create-role --role-name <app_tier_iam_role> --assume-role-
policy-document file:///tmp/TrustPolicy.json
```

- Add the IAM role created to the Instance profile:

```
aws iam add-role-to-instance-profile --role-name <app_tier_iam_role> --
instance-profile-name <app_tier_instance_profile>
```

- If doesn't exist, create an IAM managed policy for Web tier instances, and note the policy ARN:

```
aws iam create-policy --policy-name <iam_policy_name> --policy-document
file://policy
```

- Attach the IAM policy created to the App tier IAM role:

```
aws iam attach-role-policy --policy-arn <iam_policy_arn> --role-name
<app_tier_iam_role>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/iam/add-role-to-instance-profile.html
2. http://docs.aws.amazon.com/cli/latest/reference/iam/attach-role-policy.html
3. http://docs.aws.amazon.com/cli/latest/reference/iam/create-instance-profile.html
4. http://docs.aws.amazon.com/cli/latest/reference/iam/create-policy.html
5. http://docs.aws.amazon.com/cli/latest/reference/iam/get-instance-profile.html
6. http://docs.aws.amazon.com/cli/latest/reference/iam/get-policy-version.html
7. http://docs.aws.amazon.com/cli/latest/reference/iam/get-policy.html

2.3 Ensure an IAM Role for Amazon EC2 is created for Web Tier (Scored)

Profile Applicability:

- Level 1

Description:

An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

Also, a role does not have any credentials (password or access keys) associated with it. Instead, if a user is assigned to a role, access keys are created dynamically and provided to the user. You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. Applications must sign their API requests with AWS credentials. Therefore, if you are an application developer, you need a strategy for managing credentials for your applications that run on EC2 instances.

IAM Roles for EC2 allow application running within an EC2 instance assume the role applied to the instance.

Rationale:

Provides dynamic authentication credentials to which can be used with Web-Tier EC Instances once launched with the IAM Role for EC2

Audit:

Using the Amazon unified command line interface:

- List all the Web tier EC2 instances, check if and which IAM instance profile they have attached, and note the name of the instance profile:

```
aws ec2 describe-instances --filters
Name=tag:<web_tier_tag>,Values=<web_tier_tag_value> --query
"Reservations[*].Instances[*].{IamInstanceProfile:IamInstanceProfile,
InstanceId:InstanceId}"
```

- Check and note the IAM role name used by the Web tier instance profile:

```
aws iam get-instance-profile --instance-profile-name
<web_tier_instance_profile> --query "InstanceProfile.Roles[*].RoleName"
```

Remediation:

Using the Amazon unified command line interface:

- If doesn't exist, create an instance profile for Web tier instances:

```
aws iam create-instance-profile --instance-profile-name
<web_tier_instance_profile>
```

- If doesn't exist, create an IAM role for the instance profile:
    - Create a trust relationship policy document and save it locally as /tmp/TrustPolicy.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

    - Create the IAM role using the above trust policy:

```
aws iam create-role --role-name <web_tier_iam_role> --assume-role-
policy-document file:///tmp/TrustPolicy.json
```

- Add the IAM role created to the Instance profile:

```
aws iam add-role-to-instance-profile --role-name <web_tier_iam_role> --
instance-profile-name <web_tier_instance_profile>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html
2. http://docs.aws.amazon.com/cli/latest/reference/iam/create-role.html
3. http://docs.aws.amazon.com/cli/latest/reference/iam/create-instance-profile.html
4. http://docs.aws.amazon.com/cli/latest/reference/iam/add-role-to-instance-profile.html
5. http://docs.aws.amazon.com/cli/latest/reference/iam/get-instance-profile.html

2.4 Ensure an IAM Role for Amazon EC2 is created for App Tier (Scored)

Profile Applicability:

- Level 1

Description:

An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

Also, a role does not have any credentials (password or access keys) associated with it. Instead, if a user is assigned to a role, access keys are created dynamically and provided to the user. You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. Applications must sign their API requests with AWS credentials. Therefore, if you are an application developer, you need a strategy for managing credentials for your applications that run on EC2 instances.

IAM Roles for EC2 allow application running within an EC2 instance assume the role applied to the instance.

Rationale:

Provides dynamic authentication credentials to which can be used with Application Tier EC2-Instances once launched with the IAM Role for EC2

Audit:

Using the Amazon unified command line interface:

- List all the App tier EC2 instances, check if and which IAM instance profile they have attached, and note the name of the instance profile:

```
aws ec2 describe-instances --filters
Name=tag:<app_tier_tag>,Values=<app_tier_tag_value> --query
"Reservations[*].Instances[*].{IamInstanceProfile:IamInstanceProfile,
InstanceId:InstanceId}"
```

- Check and note the IAM role name used by the App tier instance profile:

```
aws iam get-instance-profile --instance-profile-name
<app_tier_instance_profile> --query "InstanceProfile.Roles[*].RoleName"
```

Remediation:

Using the Amazon unified command line interface:

- If doesn't exist, create an instance profile for App tier instances:

```
aws iam create-instance-profile --instance-profile-name
<app_tier_instance_profile>
```

- If doesn't exist, create an IAM role for the instance profile:
  - Create a trust relationship policy document and save it locally as /tmp/TrustPolicy.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

  - Create the IAM role using the above trust policy:

```
aws iam create-role --role-name <app_tier_iam_role> --assume-role-
policy-document file:///tmp/TrustPolicy.json
```

- Add the IAM role created to the Instance profile:

```
aws iam add-role-to-instance-profile --role-name <app_tier_iam_role> --
instance-profile-name <app_tier_instance_profile>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html
2. http://docs.aws.amazon.com/cli/latest/reference/iam/create-role.html
3. http://docs.aws.amazon.com/cli/latest/reference/iam/get-instance-profile.html
4. http://docs.aws.amazon.com/cli/latest/reference/iam/create-instance-profile.html
5. http://docs.aws.amazon.com/cli/latest/reference/iam/add-role-to-instance-profile.html

2.5 Ensure AutoScaling Group Launch Configuration for Web Tier is configured to use a customer created Web-Tier IAM Role (Scored)

Profile Applicability:

- Level 1

Description:

An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

Also, a role does not have any credentials (password or access keys) associated with it. Instead, if a user is assigned to a role, access keys are created dynamically and provided to the user. You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. Applications must sign their API requests with AWS credentials. Therefore, if you are an application developer, you need a strategy for managing credentials for your applications that run on EC2 instances.

IAM Roles for EC2 allow application running within an EC2 instance assume the role applied to the instance.

Rationale:

Ensures all EC2 instances within the Web-Tier auto scaling group have been launched with an IAM Role for EC2

Audit:

Using the Amazon unified command line interface:

- Check if your Web tier autoscaling group is using a launch configuration with an IAM instance profile configured:

```
aws autoscaling describe-launch-configurations --launch-configuration-names
<web_tier_launch_config> --query "LaunchConfigurations[*].IamInstanceProfile"
```

- Check and note the IAM role name used by the Web tier instance profile:

```
aws iam get-instance-profile --instance-profile-name
<web_tier_instance_profile> --query "InstanceProfile.Roles[*].RoleName"
```

Remediation:

Using the Amazon unified command line interface:

- Create new launch configuration for the Web tier using the Web tier IAM instance profile :

```
aws autoscaling create-launch-configuration --launch-configuration-name
<web_tier_launch_config> --image-id <web_tier_ami> --key-name <your_key_pair>
--security-groups <web_tier_security_group> --instance-type
<desired_instance_type> --iam-instance-profile <web_tier_instance_profile>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html
3. http://docs.aws.amazon.com/cli/latest/reference/iam/get-instance-profile.html

2.6 Ensure AutoScaling Group Launch Configuration for App Tier is configured to use an App-Tier IAM Role (Scored)

Profile Applicability:

- Level 1

Description:

An IAM role is similar to a user, in that it is an AWS identity with permission policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it.

Also, a role does not have any credentials (password or access keys) associated with it. Instead, if a user is assigned to a role, access keys are created dynamically and provided to the user. You can use roles to delegate access to users, applications, or services that don't normally have access to your AWS resources. Applications must sign their API requests with AWS credentials. Therefore, if you are an application developer, you need a strategy for managing credentials for your applications that run on EC2 instances.

IAM Roles for EC2 allow application running within an EC2 instance assume the role applied to the instance.

Rationale:

Ensures all EC2 instances within the App-Tier auto scaling group have been launched with an IAM Role for EC2

Audit:

Using the Amazon unified command line interface:

- Check if your App tier autoscaling group is using a launch configuration with an IAM instance profile configured:

```
aws autoscaling describe-launch-configurations --launch-configuration-names
<app_tier_launch_config> --query "LaunchConfigurations[*].IamInstanceProfile"
```

- Check and note the IAM role name used by the Web tier instance profile:

```
aws iam get-instance-profile --instance-profile-name
<app_tier_instance_profile> --query "InstanceProfile.Roles[*].RoleName"
```

Remediation:

Using the Amazon unified command line interface:

- Create new launch configuration for the App tier using the App tier IAM instance profile :

```
aws autoscaling create-launch-configuration --launch-configuration-name
<app_tier_launch_config> --image-id <app_tier_ami> --key-name <your_key_pair>
--security-groups <app_tier_security_group> --instance-type
<desired_instance_type> --iam-instance-profile <app_tier_instance_profile>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html
3. http://docs.aws.amazon.com/cli/latest/reference/iam/get-instance-profile.html

2.7 Ensure an IAM group for administration purposes is created (Scored)

Profile Applicability:

- Level 1

Description:

An IAM group is a collection of IAM users. You can use groups to specify permissions for a collection of users, which can make those permissions easier to manage for those users. For example, you could have a group called Admins and give that group the types of permissions that administrators typically need. Any user in that group automatically has the permissions that are assigned to the group. If a new user joins your organization and should have administrator privileges, you can assign the appropriate permissions by adding the user to that group. Similarly, if a person changes jobs in your organization, instead of editing that user's permissions, you can remove him or her from the old groups and add him or her to the appropriate new groups.

Rationale:

The IAM group will allow you to add or remove IAM users that require administrative privileges to the resources.

Audit:

Using the Amazon unified command line interface:

- List the IAM groups created for administration purposes:

```
aws iam list-groups --query "Groups[?GroupName == '<iam_admin_group_name>']"
```

Remediation:

Using the Amazon unified command line interface:

- Create a new IAM group for administration purposes:

```
aws iam create-group --group-name <iam_admin_group_name>
```

- Attach the Admin policy to the administration IAM group:

```
aws iam attach-group-policy --policy-arn <admin_policy_arn> --group-name
<iam_admin_group_name>
```

Impact:

The name of the iam admin group name should be known prior to auditing this recommendation.

For a sample admin policy arn see recommendation 2.1.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/iam/list-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/iam/create-group.html
3. http://docs.aws.amazon.com/cli/latest/reference/iam/attach-group-policy.html

2.8 Ensure an IAM policy that allows admin privileges for all services used is created (Scored)

Profile Applicability:

- Level 1

Description:

A policy is a document that formally states one or more permissions.

Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies apply only to identities (users, groups, and roles) - not resources. You must ensure that you have an IAM managed policy created with admin permissions for all the AWS services used by the application.

Rationale:

An IAM admin policy with permissions for all the AWS services used by the application must exist for administration purposes.

Audit:

Using the Amazon unified command line interface:

- List the IAM admin policies and note the policy ARN and DefaultVersionId:

```
aws iam list-policies --query "Policies[?PolicyName ==
'<admin_policy_name>']"
```

- If the policy exists, check the policy document:

```
aws iam get-policy-version --policy-arn <admin_policy_arn> --version-id
<admin_policy_version>
```

Remediation:

Using the Amazon unified command line interface:

- Create an IAM managed admin policy for all AWS services used:

```
aws iam create-policy --policy-name <admin_policy_name> --policy-document
file://policy
```

Impact:

The admin policy should be defined prior to auditing and remediating this recommendation.

For a sample admin policy see recommendation 2.1.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/iam/create-policy.html
2. http://docs.aws.amazon.com/cli/latest/reference/iam/get-policy.html
3. http://docs.aws.amazon.com/cli/latest/reference/iam/list-policies.html

2.9 Ensure SNS Topics do not Allow 'Everyone' To Publish (Scored)

Profile Applicability:

- Level 1

Description:

Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. In Amazon SNS, there are two types of clients—publishers and subscribers—also referred to as producers and consumers. Publishers communicate asynchronously with subscribers by producing and sending a message to a topic, which is a logical access point and communication channel. Subscribers (i.e., web servers, email addresses, Amazon SQS queues, AWS Lambda functions) consume or receive the message or notification over one of the supported protocols (i.e., Amazon SQS, HTTP/S, email, SMS, Lambda) when they are subscribed to the topic.

The entities who can publish messages to a SNS topic can be controlled by modifying the topic policy, and they can be:

- The topic owner
- Everyone
- Specific AWS users or resources

From the above message publisher options you should make sure that "Everyone" is not used with any SNS topic in the AWS account.

Rationale:

If a SNS topic policy allows "Everyone" to publish messages to a specific topic, this could pose a security risk as any unauthenticated entity could send malicious messages to all the topic subscribers.

Audit:

Using the Amazon unified command line interface:

- List all the SNS topics from your AWS account:

```
aws sns list-topics
```

- For each topic in step 1, list the topic attributes:

```
aws sns get-topic-attributes --topic-arn <sns_topic_arn>
```

- Check the Policy field for the presence of:

```
{
   "Effect":"Allow",
   "Principal":{
      "AWS":"*"
   },
   "Action":"SNS:Publish",
   "Resource":"<sns_topic_arn>"
}
```

Remediation:

Edit your existing policy by deleting the above snippet or replacing "Principal":{"AWS" : "*"} with "Principal":{"AWS" : "*<iam_user>*"}, and save it locally as a .json file named policy.json.
Using the Amazon unified command line interface:

- Set the new policy to the SNS topic

```
aws sns set-topic-attributes --topic-arn <sns_topic_arn> --attribute-name
Policy --attribute-value file://policy.json
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/sns/list-topics.html
2. http://docs.aws.amazon.com/cli/latest/reference/sns/get-topic-attributes.html
3. http://docs.aws.amazon.com/cli/latest/reference/sns/set-topic-attributes.html
4. http://docs.aws.amazon.com/sns/latest/dg/UsingIAMwithSNS.html#ExamplePolicies_SNS

2.10 Ensure SNS Topics do not Allow 'Everyone' To Subscribe (Scored)

Profile Applicability:

- Level 1

Description:

Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients. In Amazon SNS, there are two types of clients—publishers and subscribers—also referred to as producers and consumers. Publishers communicate asynchronously with subscribers by producing and sending a message to a topic, which is a logical access point and communication channel. Subscribers (i.e., web servers, email addresses, Amazon SQS queues, AWS Lambda functions) consume or receive the message or notification over one of the supported protocols (i.e., Amazon SQS, HTTP/S, email, SMS, Lambda) when they are subscribed to the topic.

The entities who can subscribe to a SNS topic can be controlled by modifying the topic policy, and they can be:

- The topic owner
- Everyone
- Specific AWS users or resources
- Users whose endpoint URL,protocol, email address, or ARN from a Subscriberequest match a specified value

From the above topic subscribers, you should make sure that "Everyone" is not used with any SNS topic in the AWS account.

Rationale:

If a SNS topic policy allows "Everyone" to subscribe to a specific topic, this could pose a security risk as any unauthenticated entity could subscribe and receive messages from the topic publishers, messages that should be destined only to specific, known subscribers.

Audit:

Using the Amazon unified command line interface:

- List all the SNS topics from your AWS account:

```
aws sns list-topics
```

- For each topic in step 1, list the topic attributes:

```
aws sns get-topic-attributes --topic-arn <sns_topic_arn>
```

- Check the Policy field for the presence of:

```
{
   "Effect":"Allow",
   "Principal":{
      "AWS":"*"
   },
   "Action":[
      "SNS:Subscribe",
      "SNS:Receive"
   ],
   "Resource":"<sns_topic_arn>"
}
```

Remediation:

Edit your existing policy by deleting the above snippet or replacing "Principal":{"AWS" : "*"} with "Principal":{"AWS" : "*<iam_user>*"}, and save it locally as a .json file named policy.json.
Using the Amazon unified command line interface:

- Set the new policy to the SNS topic

```
aws sns set-topic-attributes --topic-arn <sns_topic_arn> --attribute-name
Policy --attribute-value file://policy.json
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/sns/list-topics.html
2. http://docs.aws.amazon.com/cli/latest/reference/sns/get-topic-attributes.html
3. http://docs.aws.amazon.com/cli/latest/reference/sns/set-topic-attributes.html
4. http://docs.aws.amazon.com/sns/latest/dg/UsingIAMwithSNS.html#ExamplePolicies_SNS

3 Business Continuity

This section provides recommendations for application resiliency.

3.1 Ensure each Auto-Scaling Group has an associated Elastic Load Balancer (Scored)

Profile Applicability:

- Level 1

Description:

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances within a VPC.

It enables greater levels of fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to distribute application traffic across 1 or more Availability Zones within a VPC.

Elastic Load Balancing must be integrated with Auto Scaling Groups to ensure that you have availability of compute resources in the event of a failure.

Rationale:

Integrating Auto Scaling Groups with an Elastic Load Balancer will help provide high availability and back-end EC2 instance scaling.

Through Auto-Scaling Group configuration you can define:

1. minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group
2. Availability Zones / subnets used

Audit:

Using the Amazon unified command line interface:

- Identify Autoscaling Group Name and associated ELB name:

```
aws autoscaling describe-auto-scaling-groups  --query
'AutoScalingGroups[*].{ELB:LoadBalancerNames, ASGName:AutoScalingGroupName}'
```

- Identify current status of the ELB:

```
aws autoscaling describe-load-balancers --auto-scaling-group-name
<autoscaling_group_name>
```

Remediation:

Using the Amazon unified command line interface:

- List existing load balancers:

```
aws elb describe-load-balancers --query
'LoadBalancerDescriptions[*].{ELBName:LoadBalancerName}'
```

or

- Create new load balancer:

```
aws elb create-load-balancer --load-balancer-name <elb_name> --listeners
<listener_config> --subnets <application_subnet> --security-groups
<application_security_groups>
```

- Attached load balancer from previous steps to autoscaling group:

```
aws autoscaling attach-load-balancers --load-balancer-names <elb_name> --
auto-scaling-group-name <autoscaling_group_name>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/attach-load-balancers.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-load-balancers.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
5. http://docs.aws.amazon.com/cli/latest/reference/elb/create-load-balancer.html

3.2 Ensure each Auto-Scaling Group is configured for multiple Availability Zones (Scored)

Profile Applicability:

- Level 1

Description:

Auto Scaling helps maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define.

You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances or can automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

These properties can be defined within the Auto-Scaling Group configuration.

Rationale:

Ensures high availability of the application and web tiers in the event of a host or Availability Zone failure.

Audit:

Using the Amazon Unified CLI:

- List all Auto-Scaling Groups and associated Availability Zones, and ensure there is more than 1 Availability Zone assigned to the Auto-Scaling Group:

```
aws autoscaling describe-auto-scaling-groups  --query
'AutoScalingGroups[*].{AZs:AvailabilityZones, ASG:AutoScalingGroupName}'
```

Remediation:

Using the Amazon Unified CLI:

- List all the subnets and the associated Availability Zones from the application VPC:

```
aws ec2 describe-subnets --query "Subnets[?VpcId ==
'<application_vpc>'].{VPC:VpcId, Subnet:SubnetId, AZ:AvailabilityZone,
CIDR:CidrBlock}"
```

- Update AutoScaling Group to include more than 1 Availability Zones within the same VPC:

```
aws autoscaling update-auto-scaling-group --auto-scaling-group-name
<autoscaling_group_name> --availability-zones <application_az>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/update-auto-scaling-group.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html

3.3 Ensure Auto-Scaling Launch Configuration for Web-Tier is configured to use an approved Amazon Machine Image (Scored)

Profile Applicability:

- Level 1

Description:

Auto Scaling helps maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define.

You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances or can automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

These properties can be defined within the Auto-Scaling Group configuration.

Additional properties can be defined through the launch configuration such as:

- Instance Type
- Amazon Machine Image (Pre-configured Operating System Images - allows for O.S Hardening)
- IAM Role
- Security Groups

Your Organization must maintain a list of approved AMIs. Use these when creating Auto-Scaling Groups.

Rationale:

Instances within an Auto-Scaling Group are launched from an Amazon Machine Image (AMI) which itself is defined within the Launch Configuration. AMIs should be configured to follow security best practices as is defined within the CIS Benchmark for Amazon Linux or your other desired operating system.

Audit:

Using the Amazon Unified CLI:

- List the associated Launch Configuration of the Web Tier Auto-Scaling Group (note the value of "LaunchConfig" element):

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
<web_tier_autoscaling_group_name> --query
'AutoScalingGroups[*].{LaunchConfig:LaunchConfigurationName,ASG:AutoScalingGr
oupName}'
```

- Ensure actively used Launch Configuration found in the previous step is using an approved AMI from your organization's list (replace *<web_tier_launch_config>* with the Launch Configuration previously found):

```
aws autoscaling describe-launch-configurations --launch-configuration-names
<web_tier_launch_config> --query
'LaunchConfigurations[*].{LaunchConfig:LaunchConfigurationName, AMI:ImageId,
InstanceType:InstanceType}'
```

Remediation:

Using the Amazon unified command line interface:

- Create new launch configuration for the Web tier using the approved Web tier AMI from your organization's list:

```
aws autoscaling create-launch-configuration --launch-configuration-name
<new_web_tier_launch_config> --image-id <web_tier_ami> --key-name
<your_key_pair> --security-groups <web_tier_security_group> --instance-type
<desired_instance_type> --iam-instance-profile <web_tier_instance_profile>
```

Impact:

A list of approved AMIs must be maintained by the organization.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html

3.4 Ensure Auto-Scaling Launch Configuration for App-Tier is configured to use an approved Amazon Machine Image (Scored)

Profile Applicability:

- Level 1

Description:

Auto Scaling helps maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define.

You should use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances or can automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

These properties can be defined within the Auto-Scaling Group configuration.

Additional properties can be defined through the launch configuration such as:

- Instance Type
- Amazon Machine Image (Pre-configured Operating System Images - allows for O.S Hardening)
- IAM Role
- Security Groups

Your organization must maintain a list of approved AMIs. Use these when creating Auto-Scaling Groups.

Rationale:

Instances within an Auto-Scaling Group are launched from an Amazon Machine Image (AMI) which itself is defined within the Launch Configuration. The AMI should be configured to follow security best practices as is defined within the CIS Benchmark for Amazon Linux or your other desired operating system.

Audit:

Using the Amazon Unified CLI:

- List the associated Launch Configuration of the App Tier Auto-Scaling Group (note the value of "LaunchConfig" element):

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
<app_tier_autoscaling_group_name> --query
'AutoScalingGroups[*].{LaunchConfig:LaunchConfigurationName,ASG:AutoScalingGr
oupName}'
```

- Ensure actively used Launch Configuration found in the previous step is using an approved AMI from your organization's list (replace *<app_tier_launch_config>* with the Launch Configuration previously found):

```
aws autoscaling describe-launch-configurations --launch-configuration-names
<app_tier_launch_config> --query
'LaunchConfigurations[*].{LaunchConfig:LaunchConfigurationName, AMI:ImageId,
InstanceType:InstanceType}'
```

Remediation:

Using the Amazon unified command line interface:

- Create new launch configuration for the App tier using the approved App tier AMI from your organization's list:

```
aws autoscaling create-launch-configuration --launch-configuration-name
<new_app_tier_launch_config> --image-id <app_tier_ami> --key-name
<your_key_pair> --security-groups <app_tier_security_group> --instance-type
<desired_instance_type> --iam-instance-profile <app_tier_instance_profile>
```

Impact:

A list of approved AMIs must be maintained by the organization. These AMIs must be hardened to comply with the best practices of the organization.

Default Value:

The prescribed value is not the default value.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html

3.5 Ensure Relational Database Service is Multi-AZ Enabled (Scored)

Profile Applicability:

- Level 1

Description:

Amazon Relational Database Service (RDS) is a managed relational database service which handles routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.

There are 6 database engines available for customer to run their database workloads on:

- Amazon Aurora (MySQL Compatible)
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- PostgreSQL

Rationale:

Provides AWS managed high availability of the Database Tier across 2 availability zones within a region through asynchronous replication at the data layer.

Audit:

Using the Amazon unified command line interface:

- Check if your application DB instances are Multi-AZ enabled:

```
aws rds describe-db-instances --filters
Name=tag:<data_tier_tag>,Values=<data_tier_tag_value> --query
"DBInstances[*].{MultiAZ:MultiAZ, DBInstanceIdentifier:DBInstanceIdentifier}"
```

Remediation:

Using the Amazon unified command line interface:

- Modify each no-multi-az DB instance, and make it Multi-AZ enabled:

```
aws rds modify-db-instance --db-instance-identifier <your_db_instance> --
multi-az
```

Impact:

For data transferred between an Amazon EC2 instance and Amazon RDS DB Instance in different Availability Zones of the same Region, Amazon EC2 Regional Data Transfer charges apply on both sides of transfer.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-instances.html
2. http://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-instance.html

3.6 Ensure Relational Database Service Instances have Auto Minor Version Upgrade Enabled (Scored)

Profile Applicability:

- Level 1

Description:

Amazon Relational Database Service (RDS) is a managed relational database service which handles routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.

There are 6 database engines available for customer to run their database workloads on:

- Amazon Aurora (MySQL Compatible)
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- PostgreSQL

If the database engine used by your application supports it, ensure that the RDS Instances have Auto Minor Version Upgrade Enabled.

Rationale:

Ensures automated patch management is in place on the RDS instance to ensure the database engine has all the latest patches applied.

Audit:

Using the Amazon unified command line interface:

- Check if your application DB instances have Auto Minor Version Upgrade enabled:

```
aws rds describe-db-instances --filters
Name=tag:<data_tier_tag>,Values=<data_tier_tag_value> --query
"DBInstances[*].{AutoMinorVersionUpgrade:AutoMinorVersionUpgrade,
DBInstanceIdentifier:DBInstanceIdentifier}"
```

Remediation:

Using the Amazon unified command line interface:

- Modify each DB instance with auto-minor-version-upgrade set to False, and enable auto-minor-version-upgrade:

```
aws rds modify-db-instance --db-instance-identifier <your_db_instance> --
auto-minor-version-upgrade
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-instances.html
2. http://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-instance.html

3.8 Ensure Relational Database Service backup retention policy is set (Scored)

Profile Applicability:

- Level 1

Description:

Amazon Relational Database Service (RDS) is a managed relational database service which handles routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.

There are 6 database engines available for customer to run their database workloads on:

- Amazon Aurora (MySQL Compatible)
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- PostgreSQL

Rationale:

Provides a managed backup function of the RDS Database, it is possible to define the backup window and retention period of the backup. Each customer should have a retention policy set for the type of data being stored. Recommend setting this to at least 7.

Possible values are from 0 to 35 days.

Audit:

Using the Amazon unified command line interface:

- Check if your application DB instances have a Backup Retention Period set (0 = there is no backup retention in place, 7 = there are 7 daily backups retained):

```
aws rds describe-db-instances --filters
Name=tag:<data_tier_tag>,Values=<data_tier_tag_value> --query
"DBInstances[*].{BackupRetentionPeriod:BackupRetentionPeriod,
DBInstanceIdentifier:DBInstanceIdentifier}"
```

Remediation:

Using the Amazon unified command line interface:

- Modify each DB instance with Backup Retention Period of 0, and set a desired Backup Retention Period in days (recommended value = 7):

```
aws rds modify-db-instance --db-instance-identifier <your_db_instance> --
backup-retention-period <backup_retention_period>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-instances.html
2. http://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-instance.html

3.9 Ensure Web Tier Elastic Load Balancer has application layer Health Check Configured (Scored)

Profile Applicability:

- Level 1

Description:

By default, an Auto-Scaling Group periodically uses the results of the EC2 instance status checks to determine the health status of each instance. If an instance fails the EC2 instance status checks, Auto-Scaling marks the instance as unhealthy and replaces the instance.

However, if you have attached one or more Elastic Load Balancing (ELB) load balancers to your Auto-Scaling Group and the instance fails the ELB health checks, Auto-Scaling does not replace the instance.

Amazon ELB will periodically sends pings, attempts connections, or sends requests to test the EC2 instances, these tests are called health checks.

The status of the instances that are healthy at the time of the health check is InService.

The status of any instances that are unhealthy at the time of the health check is OutOfService.

The load balancer performs health checks on all registered instances, whether the instance is in a healthy state or an unhealthy state. The load balancer routes requests only to the healthy instances. When the load balancer determines that an instance is unhealthy, it stops routing requests to that instance. The load balancer resumes routing requests to the instance when it has been restored to a healthy state

Rationale:

Ensures availability of back-end EC2 instances associated with an Amazon ELB through application layer health check (ex: http) instead of TCP health checks.

Audit:

Using the Amazon unified CLI:

- Identify if healthcheck is in place on the Web tier ELB:

```
aws elb describe-load-balancers --load-balancer-names <web_tier_elb> --query
'LoadBalancerDescriptions[*].{ELBName:LoadBalancerName,HealthCheck:HealthChec
k}'
```

Remediation:

Using the Amazon unified CLI:

- Create a JSON file containing the attributes you want to modify and save it locally as /tmp/ELBhealthcheck.json:

```
{
  "Target": "<string>",
  "Interval": <integer>,
  "Timeout": <integer>,
  "UnhealthyThreshold": <integer>,
  "HealthyThreshold": <integer>
}
```

- Modify Web tier ELB to include appropriate health check:

```
aws elb configure-health-check --load-balancer-name <web_tier_elb> --health-
check file:///tmp/ELBhealthcheck.json
```

Impact:

Ensure appropriate ACLs are in place between ELB and EC2 backend instances in order for healthchecks to work correctly.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/elb/configure-health-check.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html

3.10 Ensure App Tier Elastic Load Balancer has application layer Health Check Configured (Scored)

Profile Applicability:

- Level 1

Description:

By default, an Auto-Scaling Group periodically uses the results of the EC2 instance status checks to determine the health status of each instance. If an instance fails the EC2 instance status checks, Auto-Scaling marks the instance as unhealthy and replaces the instance.

However, if you have attached one or more Elastic Load Balancing (ELB) load balancers to your Auto-Scaling Group and the instance fails the ELB health checks, Auto-Scaling does not replace the instance.

Amazon ELB will periodically sends pings, attempts connections, or sends requests to test the EC2 instances, these tests are called health checks.

The status of the instances that are healthy at the time of the health check is InService.

The status of any instances that are unhealthy at the time of the health check is OutOfService.

The load balancer performs health checks on all registered instances, whether the instance is in a healthy state or an unhealthy state. The load balancer routes requests only to the healthy instances. When the load balancer determines that an instance is unhealthy, it stops routing requests to that instance. The load balancer resumes routing requests to the instance when it has been restored to a healthy state

Rationale:

Ensures availability of back-end EC2 instances associated with an Amazon ELB through application layer health check (ex: http) instead of TCP health checks.

Audit:

Using the Amazon unified CLI:

- Identify if healthcheck is in place on the App tier ELB:

```
aws elb describe-load-balancers --load-balancer-names <app_tier_elb> --query
'LoadBalancerDescriptions[*].{ELBName:LoadBalancerName,HealthCheck:HealthChec
k}'
```

Remediation:

Using the Amazon unified CLI:

- Create a JSON file containing the attributes you want to modify and save it locally as /tmp/ELBhealthcheck.json:

```
{
  "Target": "<string>",
  "Interval": <integer>,
  "Timeout": <integer>,
  "UnhealthyThreshold": <integer>,
  "HealthyThreshold": <integer>
}
```

- Modify App tier ELB to include appropriate health check:

```
aws elb configure-health-check --load-balancer-name <app_tier_elb> --health-
check file:///tmp/ELBhealthcheck.json
```

Impact:

Ensure appropriate ACLs are in place between ELB and EC2 backend instances in order for healthchecks to work correctly.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/elb/configure-health-check.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html

3.11 Ensure S3 buckets have versioning enabled (Scored)

Profile Applicability:

- Level 1

Description:

Amazon S3 can further protect your data using versioning. Be sure to enable this feature.

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. By default, requests retrieve the most recently written version. You can retrieve older versions of an object by specifying a version of the object in a request.

Rationale:

Versioning-enabled buckets enable you to recover objects from accidental deletion or overwrite. For example:

- If you delete an object, instead of removing it permanently, Amazon S3 inserts a delete marker, which becomes the current object version. You can always restore the previous version.
- If you overwrite an object, it results in a new object version in the bucket. You can always restore the previous version.

Audit:

Using the Amazon unified command line interface:

- List all the S3 buckets from the AWS account:

```
aws s3api list-buckets --output table
```

- For each of the S3 buckets check if versioning is enabled, if the output is empty versioning is disabled:

```
aws s3api get-bucket-versioning --bucket <s3_bucket_name>
```

Remediation:

Using the Amazon unified command line interface:

- Enable versioning for all the S3 buckets that does not have this feature enabled

```
aws s3api put-bucket-versioning --bucket <s3_bucket_name> --versioning-
configuration Status=Enabled
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/s3api/get-bucket-versioning.html
2. http://docs.aws.amazon.com/cli/latest/reference/s3api/put-bucket-versioning.html
3. http://docs.aws.amazon.com/cli/latest/reference/s3api/list-buckets.html
4. http://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html

3.12 Configure HTTP to HTTPS Redirects with a CloudFront Viewer Protocol Policy (Scored)

Profile Applicability:

- Level 2

Description:

Configure the Viewer Protocol Policy for your CloudFront cache to redirect HTTP requests to HTTPS requests or to require that viewers use only the HTTPS protocol to access your objects in the CloudFront cache. You should also configure one or more cache behaviors in the same distribution to allow both HTTP and HTTPS, so you can require HTTPS for some objects but not for others.

In order to use HTTPS, a SSL\TLS certificate must be attached.

This depends on your data classification policy and needs to be configured according to your encryption policy.

Rationale:

To ensure that objects are encrypted from edge locations to viewers using HTTP or HTTPS depending on your data classification and encryption policies, use only HTTPS.

Audit:

Using the Amazon unified command line interface:

- Check the "ViewerProtocolPolicy" used by application Cloudfront distributions (it should be set as redirect-to-https):

```
aws cloudfront list-distributions --query "DistributionList.Items[*].{Id:Id,
ViewerProtocolPolicy:DefaultCacheBehavior.ViewerProtocolPolicy}"
```

Remediation:

Using the Amazon unified command line interface:

- For configuring "ViewerProtocolPolicy" first save locally the current distribution config:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "DistributionConfig" > /tmp/cf-distribution.json
```

- Edit and replace "ViewerProtocolPolicy" element in /tmp/cf-distribution.json with the below section:

```
"ViewerProtocolPolicy": "redirect-to-https",
```

- Retrieve the current ETag of your CloudFront distribution:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "ETag"
```

- Update the CloudFront distribution using the edited config and the above Etag:

```
aws cloudfront update-distribution --id <application_cfn_distribution_id> --
distribution-config file:///tmp/cf-distribution.json --if-match
<application_cfn_distribution_etag>
```

References:

1. http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/SecureConnections.html
2. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/update-distribution.html
3. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/get-streaming-distribution-config.html
4. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/list-distributions.html

3.13 Ensure all CloudFront Distributions require HTTPS between CloudFront and your Web-Tier ELB origin (Scored)

Profile Applicability:

- Level 2

Description:

Configure the Origin Protocol Policy for the Web tier ELB origin either to require that CloudFront fetches objects from your origin by using HTTPS or to require that CloudFront uses the protocol that the viewer used to request the objects. For example, if you choose Match Viewer for the Origin Protocol Policy and the viewer uses HTTPS to request an object from CloudFront, CloudFront also uses HTTPS to forward the request to your origin.

In order to use HTTPS, an SSL\TLS certificate must be attached.

Rationale:

To ensure that objects are encrypted from edge locations to the Web-Tier ELB origin according to the data classification policy, use Match Viewer.

Audit:

Using the Amazon unified command line interface:

- Check the "OriginProtocolPolicy" used by application Cloudfront distributions (it should be set as https-only):

```
aws cloudfront list-distributions --query "DistributionList.Items[*].{Id:Id,
OriginProtocolPolicy:Origins.Items[*].CustomOriginConfig.OriginProtocolPolicy
}"
```

Remediation:

Using the Amazon unified command line interface:

- For configuring "OriginProtocolPolicy"first save locally the current distribution config:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "DistributionConfig" > /tmp/cf-distribution.json
```

- Edit and replace "OriginProtocolPolicy"element in /tmp/cf-distribution.json with the below section:

```
"OriginProtocolPolicy": "https-only",
```

- Retrieve the current ETag of your CloudFront distribution:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "ETag"
```

- Update the CloudFront distribution using the edited config and the above Etag:

```
aws cloudfront update-distribution --id <application_cfn_distribution_id> --
distribution-config file:///tmp/cf-distribution.json --if-match
<application_cfn_distribution_etag>
```

References:

1. http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/SecureConnections.html
2. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/update-distribution.html
3. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/get-streaming-distribution-config.html
4. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/list-distributions.html

3.14 Ensure Web Tier Auto-Scaling Group has an associated Elastic Load Balancer (Scored)

Profile Applicability:

- Level 1

Description:

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances within a VPC.

It enables greater levels of fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to distribute application traffic across 1 or more Availability Zones within a VPC.

Elastic Load Balancing must be integrated with Auto Scaling Groups to ensure that you have availability of compute resources in the event of a failure.

Rationale:

Integrating Auto Scaling Groups with an Elastic Load Balancer will help provide high availability and back-end EC2 instance scaling.

Through Auto-Scaling Group configuration you can define:

1. minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group
2. Availability Zones / subnets used

Audit:

Using the Amazon unified command line interface:

- Identify Autoscaling Group Name and associated ELB name:

```
aws autoscaling describe-auto-scaling-groups  --query
'AutoScalingGroups[*].{ELB:LoadBalancerNames, ASGName:AutoScalingGroupName}'
```

- Identify current status of the ELB:

```
aws autoscaling describe-load-balancers --auto-scaling-group-name
<web_tier_autoscaling_group_name>
```

Remediation:

Using the Amazon unified command line interface:

- List existing load balancers:

```
aws elb describe-load-balancers --query
'LoadBalancerDescriptions[*].{ELBName:LoadBalancerName}'
```

or

- Create new load balancer:

```
aws elb create-load-balancer --load-balancer-name <web_tier_elb> --listeners
<listener_config> --subnets <web_tier_elb_subnet1> <web_tier_elb_subnet2> --
security-groups <web_tier_elb_security_group>
```

- Attached load balancer from previous steps to autoscaling group:

```
aws autoscaling attach-load-balancers --load-balancer-names <web_tier_elb> --
auto-scaling-group-name <web_tier_autoscaling_group_name>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/attach-load-balancers.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-load-balancers.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
5. http://docs.aws.amazon.com/cli/latest/reference/elb/create-load-balancer.html

3.15 Ensure App Tier Auto-Scaling Group has an associated Elastic Load Balancer (Scored)

Profile Applicability:

- Level 1

Description:

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances within a VPC.

It enables greater levels of fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to distribute application traffic across 1 or more Availability Zones within a VPC.

Elastic Load Balancing must be integrated with Auto Scaling Groups to ensure that you have availability of compute resources in the event of a failure.

Rationale:

Integrating Auto Scaling Groups with an Elastic Load Balancer will help provide high availability and back-end EC2 instance scaling.

Through Auto-Scaling Group configuration you can define:

1. minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group
2. Availability Zones / subnets used

Audit:

Using the Amazon unified command line interface:

- Identify Autoscaling Group Name and associated ELB name:

```
aws autoscaling describe-auto-scaling-groups  --query
'AutoScalingGroups[*].{ELB:LoadBalancerNames, ASGName:AutoScalingGroupName}'
```

- Identify current status of the ELB:

```
aws autoscaling describe-load-balancers --auto-scaling-group-name
<app_tier_autoscaling_group_name>
```

Remediation:

Using the Amazon unified command line interface:

- List existing load balancers:

```
aws elb describe-load-balancers --query
'LoadBalancerDescriptions[*].{ELBName:LoadBalancerName}'
```

or

- Create new load balancer:

```
aws elb create-load-balancer --load-balancer-name <app_tier_elb> --scheme
internal --listeners <listener_config> --subnets <app_tier_subnet1>
<app_tier_subnet2> --security-groups <app_tier_elb_security_group>
```

- Attached load balancer from previous steps to autoscaling group:

```
aws autoscaling attach-load-balancers --load-balancer-names <app_tier_elb> --
auto-scaling-group-name <app_tier_autoscaling_group_name>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/attach-load-balancers.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-load-balancers.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
5. http://docs.aws.amazon.com/cli/latest/reference/elb/create-load-balancer.html

4 Event Monitoring and Response

This section contains recommendations for detecting and responding to events.

4.1 Ensure a SNS topic is created for sending out notifications from Cloudtwatch Alarms and Auto-Scaling Groups (Scored)

Profile Applicability:

- Level 1

Description:

For the Cloudwatch alarms and Auto-Scaling Groups to be able to send out notifications, a SNS topic should be created.

Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.

When using Amazon SNS, you (as the owner) create a topic and control access to it by defining policies that determine which publishers and subscribers can communicate with the topic.

Rationale:

Cloudwatch alarms and certain actions inside Auto-Scaling Groups needs to be sent out to administrators, in order to be acted upon.

Audit:

Using the Amazon unified CLI:

- List all cloudwatch alarms and filter by alarm action (review "AlarmActions": [<sns_topic_arn>]):

```
aws cloudwatch describe-alarms --query 'MetricAlarms[*].{AlarmName:AlarmName,
AlarmActions:AlarmActions, Dimensions:Dimensions}'
```

- List SNS topic attributes:

```
aws sns list-topic-attributes --topic-arn <sns_topic_arn>
```

- List SNS topic subscriptions (endpoint which receives messages captured by the SNS topic):

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

Remediation:

Using the Amazon unified CLI:

- Create a new topic, and note the topic-arn value:

```
aws sns create-topic --name <sns_topic_name>
```

- Create a subscription to the new topic:

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/sns/list-subscriptions-by-topic.html
2. http://docs.aws.amazon.com/cli/latest/reference/sns/get-topic-attributes.html
3. http://docs.aws.amazon.com/cli/latest/reference/sns/create-topic.html
4. http://docs.aws.amazon.com/cli/latest/reference/sns/subscribe.html

4.2 Ensure a SNS topic is created for sending out notifications from RDS events (Scored)

Profile Applicability:

- Level 1

Description:

For the RDS event subscriptions to be able to send out notifications, a SNS topic should be created.

Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribing endpoints or clients.

When using Amazon SNS, you (as the owner) create a topic and control access to it by defining policies that determine which publishers and subscribers can communicate with the topic.

Rationale:

RDS events generaged through defined RDS event subscriptions needs to be sent out to administrators, in order to be acted upon.

Audit:

Using the Amazon unified CLI:

- List all RDS event subscriptions in order to capture the topic-arn:

```
aws rds describe-event-subscriptions --query
'EventSubscriptionsList[*].{SourceType:SourceType,
SourceIdsList:SourceIdsList, EventCategoriesList:EventCategoriesList}'
```

- List SNS topic attributes:

```
aws sns list-topic-attributes --topic-arn <sns_topic_arn>
```

- List SNS topic subscriptions (endpoint which receives messages captured by the SNS topic):

```
aws sns list-subscriptions-by-topic --topic-arn <sns_topic_arn>
```

Remediation:

Using the Amazon unified CLI:

- Create a new topic, and note the topic-arn value:

```
aws sns create-topic --name <sns_topic_name>
```

- Create a subscription to the new topic:

```
aws sns subscribe --topic-arn <sns_topic_arn> --protocol <protocol_for_sns> --notification-endpoint <sns_subscription_endpoints>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/sns/list-subscriptions-by-topic.html
2. http://docs.aws.amazon.com/cli/latest/reference/sns/get-topic-attributes.html
3. http://docs.aws.amazon.com/cli/latest/reference/sns/create-topic.html
4. http://docs.aws.amazon.com/cli/latest/reference/sns/subscribe.html

4.3 Ensure RDS event subscriptions are enabled for Instance level events (Scored)

Profile Applicability:

- Level 1

Description:

AWS Relational Database Services offers customers a managed database engine solution for hosting customer created databases which can allow for a reduction in operational burden on customers.

RDS event subscriptions provide notification of selected event changes at Data Base engine level such as:

- Deletion
- Failure
- Failover
- Low Storage
- Maintentance

Rationale:

Event subscriptions are designed to provide incident notification of events which may affect the availability of a RDS database instance.

Audit:

Using the Amazon unified CLI:

- List all present event subscriptions and review the value of "db-instance" associated with "SourceType" element:

```
aws rds describe-event-subscriptions --query
'EventSubscriptionsList[*].{SourceType:SourceType,
SourceIdsList:SourceIdsList, EventCategoriesList:EventCategoriesList}'
```

-
    - "EventCategoriesList" will list all event categories which will be reported on
    - "SourceIdsList" will list all RDS DB instances included (null=all instances)

Remediation:

Using the Amazon unified CLI:

- Create a new event subscription for DB instance level events:

```
aws rds create-event-subscription --subscription-name
<rds_event_subscription> --sns-topic-arn <sns_topic_arn> --source-type db-
instance --event-categories <rds_events> --source-ids <events_source_ids> --
enabled
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-event-subscriptions.html
2. http://docs.aws.amazon.com/cli/latest/reference/rds/create-event-subscription.html
3. http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_Events.html#USER_Events.Messages

4.4 Ensure RDS event subscriptions are enabled for DB security groups (Scored)

Profile Applicability:

- Level 1

Description:

AWS Relational Database Services offers customers a managed database engine solution for hosting customer created databases which can allow for a reduction in operational burden on customers.

RDS event subscriptions provide notification of selected event changes at a DB security group level.

Rationale:

Event subscriptions are designed to provide incident notification of events which may affect the network availability of the RDS instance.

Audit:

Using the Amazon unified CLI:

- List all present event subscriptions and review the value of "db-security-group" associated with "SourceType" element:

```
aws rds describe-event-subscriptions --query
'EventSubscriptionsList[*].{SourceType:SourceType,
SourceIdsList:SourceIdsList, EventCategoriesList:EventCategoriesList}'
```

-
    - "EventCategoriesList" will list all event categories which will be reported on
    - "SourceIdsList" will list all RDS DB instances included (null=all instances)

Remediation:

Using the Amazon unified CLI:

- Create a new event subscription for DB Security Group events:

```
aws rds create-event-subscription --subscription-name
<rds_event_subscription> --sns-topic-arn <sns_topic_arn> --source-type db-
security-group --event-categories <rds_events> --source-ids
<events_source_ids> --enabled
```

4.6 Ensure that a log metric filter for the Cloudwatch group assigned to the "VPC Flow Logs" is created (Scored)

Profile Applicability:

- Level 1

Description:

This recommendation builds upon the Foundation benchmark recommendation: "Ensure VPC Flow Logging is Enabled in all Applicable Regions"

VPC flow logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data is stored using Amazon CloudWatch Logs. VPC flow logs can capture accepted traffic, rejected traffic, or all traffic.

Metric filters can be used to express how the service would extract metric observations from ingested events and transform them to data points in a CloudWatch metric. Metric filters are assigned to log groups, and all of the filters assigned to a log group are applied to their log streams.

A metric filter should be created for counting how many IP packets are rejected in the VPC flow logs.

Rationale:

For being able to quantify and have an accurate image of the rejected IP packets in the VPC, a metric filter must be assigned to the Cloudwatch log group created by the "VPC Flow Logs".

Audit:

Using the Amazon unified command line interface to check if the log metric filter :

```
aws logs describe-metric-filters --region <application_region>  --log-group-name <vpc_flow_log_group_name>
```

Remediation:

Using the Amazon unified command line interface:

- Create a metric filter for the Cloudwatch Log group assigned to the "VPC Flow Logs":

```
aws logs put-metric-filter --log-group-name <vpc_flow_log_group_name> --
filter-name <vpc_flow_log_filter_name> --filter-pattern "{ $.errorCode =
\"AccessDenied\" }" --metric-transformations
metricName=<vpc_flow_log_metric_name>,metricNamespace=LogMetrics,metricValue=
1
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/logs/filter-log-events.html
2. http://docs.aws.amazon.com/cli/latest/reference/logs/put-metric-filter.html
3. http://docs.aws.amazon.com/cli/latest/reference/logs/describe-metric-filters.html

4.7 Ensure that a Cloudtwach Alarm is created for the "VPC Flow Logs" metric filter, and an Alarm Action is configured (Scored)

Profile Applicability:

- Level 1

Description:

A Cloudwatch alarm watches a single metric over a time period you specify, and performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic.

The Cloudtwatch Alarm will trigger a notification being sent to the administrators every time the "REJECT packets" specified threshold is reached. The alarm should be created for the "VPC Flow Logs" metric, and the action should have a SNS topic configured.

Rationale:

For the administrators subscribed to a SNS topic to be able to receive notifications when IP packets are rejected inside the VPC, a Cloudwatch alarm must be configured for the "VPC Flow Logs" metric.

Audit:

Using the Amazon unified command line interface:

- List all the Cloudwatch alarms configured for the VPC Flow Logs metric, and check if an Alarm Action is configured:

```
aws cloudwatch describe-alarms --query "MetricAlarms[?MetricName ==
'<vpc_flow_log_metric_name>'].{MetricName:MetricName,
AlarmActions:AlarmActions,AlarmName:AlarmName}"
```

Remediation:

Using the Amazon unified command line interface:

- Create a Cloudwatch alarm for the VPC Flow Logs metric, and configure an Alarm Action:

```
aws cloudwatch put-metric-alarm --alarm-name <vpc_flow_log_alarm_name> --
alarm-actions <sns_topic_arn> --metric-name <vpc_flow_log_metric_name> --
namespace LogMetrics --statistic <desired_statistic> --period
<desired_period> --evaluation-periods <desired_evaluation_periods> --
threshold <desired_threshold> --comparison-operator <desired_operator>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/cloudwatch/put-metric-alarm.html
2. http://docs.aws.amazon.com/cli/latest/reference/cloudwatch/describe-alarms.html

4.8 Ensure Billing Alerts are enabled for increments of X spend (Scored)

Profile Applicability:

- Level 1

Description:

AWS Billing and Cost Management is the service that you use to pay your AWS bill, monitor your usage, and budget your costs. The Billing and Cost Management service provides features that you can use to estimate and plan your AWS costs, receive alerts if your costs exceed a threshold that you set, assess your biggest investments in AWS resources. Once all resources are tagged, it becomes possible to perform detailed billing analysis on a per tag basis.

Rationale:

Provides billing notifications based on per-determined dollar value intervals. Billing alerts help prevent unexpected spend increases which may be due to:

1. Higher than normal traffic load - resulting in a larger number of instances per auto-scaling group
2. Shadow I.I. - resources which have been created but are no longer in use
3. Unauthorized account or instance usage

Audit:

Check if billing alerts are enabled.

1. Sign in to the AWS Management Console and open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home#/.
2. On the navigation pane, choose Preferences.
3. Make sure that Receive Billing Alerts check box.

The second step is using the Amazon unified command line interface to check if the alarm for the AWS/Billing namespace is created:

```
aws cloudwatch describe-alarms --region us-east-1 --alarm-names
<billing_alarm_name>
```

Remediation:

Before you create a billing alarm, you must enable billing alerts. You need to do this only once.

1. Sign in to the AWS Management Console and open the Billing and Cost Management console at https://console.aws.amazon.com/billing/home#/.
2. On the navigation pane, choose Preferences.
3. Select the Receive Billing Alerts check box.
4. Choose Save preferences.

The second step is using the Amazon unified command line interface to create the billing alarm based on the EstimatedCharges metric:

1. Using the Amazon unified command line interface, list all available Amazon CloudWatch metrics for the AWS services that you're using.:

```
aws cloudwatch list-metrics
```

2. In the list of metrics, review the billing metrics that have the AWS/Billing namespace. These are the billing metrics that you can use to create a billing alarm.

3. Using the Amazon unified command line interface:

```
aws cloudwatch put-metric-alarm --alarm-name <billing_alarm_name> --
comparison-operator GreaterThanOrEqualToThreshold --evaluation-periods 1 --
metric-name <estimated_charges> --namespace AWS/Billing --dimensions
Name=Currency,Value=USD --period 21600 --statistic Maximum --threshold
<integer> --actions-enabled --alarm-actions <sns_topic_arn>
```

References:

1. http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/monitor_estimated_charges_with_cloudwatch.html
2. http://docs.aws.amazon.com/cli/latest/reference/cloudwatch/describe-alarms.html
3. http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/free-tier-alarms.html

5 Audit and Logging

This section provides recommendations that supporting auditing the application stack.

5.1 Ensure all resources are correctly tagged (Not Scored)

Profile Applicability:

- Level 1

Description:

Tags enable customers to categorize AWS resources in different ways, for example, by purpose, owner, or environment.

Each tag consists of a key and an optional value, both of which customer's define.

You should define a set of tags for the following items to help you track each instance's owner and operating environment, cost center, and other items.

1. Amazon EC2 instances
2. ELB
3. EBS Volumes
4. S3 Buckets

A resource may have up to 10 tags associated with key & value such as:

- Key=tier, Value=app
- Key=environment, Value=production
- Key=costcenter, Value=sales

Rationale:

Tagging creates a unique set of identifiers which can be applied to AWS resources such as EC2.

Tagging enables the following:

- detailed billing analysis
- access management through AWS IAM policies
- asset management through AWS Config Rules
- AWS Inspector assessment groups

Audit:

Verify tags are being used.

Remediation:

Tag all your 3 tier Web Application resources based on their tier membership (Web, App, Data), and your organizational requirements.

References:

1. https://aws.amazon.com/blogs/aws/resource-groups-and-tagging/
2. http://d0.awsstatic.com/whitepapers/compliance/AWS_Anitian_Wookbook_PCI_Cloud_Compliance.pdf

5.2 Ensure AWS Elastic Load Balancer logging is enabled (Scored)

Profile Applicability:

- Level 1

Description:

Elastic Load Balancing automatically distributes incoming application traffic across multiple Amazon EC2 instances in the a VPC. It enables you to achieve greater levels of fault tolerance in your applications, seamlessly providing the required amount of load balancing capacity needed to distribute application traffic.

Rationale:

AWS Elastic Load Balancers (ELBs) can record all incoming request sent to the load balancer and store within logs stored on S3. This allows for diagnosing application failures and analyzing web traffic and security analysis of incoming traffic

Audit:

Via Amazon unified CLI:

- List all ELB's:

```
aws elb describe-load-balancers --query
'LoadBalancerDescriptions[*].{LoadBalancerName:LoadBalancerName}'
```

- Confirm correct ELB is selected by review ELB tags:

```
aws elb describe-tags --load-balancer-names <elb_name>
```

- Review selected ELB attributes and ensure Access Log is enabled:

```
aws elb describe-load-balancer-attributes --load-balancer-name <elb_name>
```

Remediation:

Using the Amazon unified CLI:

- Create a JSON file containing the attributes you want to modify and save it locally as /tmp/ElbLogs.json:

```
{
"AccessLog": {
    "Enabled": true,
    "S3BucketName": "string",
    "EmitInterval": integer,
    "S3BucketPrefix": "string"
  }
}
```

- Update the Load Balancer attributes:

```
aws elb modify-load-balancer-attributes --load-balancer-name <elb_name> --
load-balancer-attributes file:///tmp/ElbLogs.json
```

References:

1. https://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/enable-access-logs.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
3. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancer-attributes.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-tags.html
5. http://docs.aws.amazon.com/cli/latest/reference/elb/modify-load-balancer-attributes.html

5.3 Ensure AWS Cloudfront Logging is enabled (Scored)

Profile Applicability:

- Level 1

Description:

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, for example, .html, .css, .php, image, and media files, to end users. CloudFront delivers your content through a worldwide network of edge locations. When an end user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency, so content is delivered with the best possible performance. If the content is already in that edge location, CloudFront delivers it immediately. If the content is not currently in that edge location, CloudFront retrieves it from a customer defined Origin, such as AWS S3, AWS ELB or EC2.

Rationale:

Access logs are activity records that show you detailed information about each request made for your content. These logs can be used for security analysis regarding vulnerability and availability threats

Audit:

Using the Amazon unified CLI:

- List all Cloudfront distributions and identify your application Distribution and Origin IDs. Note the "Id:" element:

```
aws cloudfront list-distributions
```

- Using the distribution Id from the previous step display the distribution settings, and check if the "Enabled" element is True or False:

```
aws cloudfront get-distribution-config --id
<application_cfn_distribution_id>--query "DistributionConfig.Logging"
```

Remediation:

Using the Amazon unified command line interface:

- For enabling logging first save locally the current distribution config:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "DistributionConfig" > /tmp/cf-distribution.json
```

- Edit and replace "Logging" element in /tmp/cf-distribution.json with the below section:

```
"Logging": {
      "Bucket": "<s3_bucket_name>",
      "Prefix": "cloudfrontlogs",
      "Enabled": true,
      "IncludeCookies": false
  },
```

- Retrieve the current ETag of your CloudFront distribution:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "ETag"
```

- Update the CloudFront distribution using the edited config and the above Etag:

```
aws cloudfront update-distribution --id <application_cfn_distribution_id> --
distribution-config file:///tmp/cf-distribution.json --if-match
<application_cfn_distribution_etag>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/update-distribution.html
2. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/list-distributions.html
3. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/get-distribution-config.html

5.4 Ensure Cloudwatch Log Group is created for Web Tier (Scored)

Profile Applicability:

- Level 1

Description:

AWS CloudWatch Log groups define groups of log streams that share the same retention, monitoring, and access control settings. Each log stream has to belong to one log group.

Note:

- You can also use any third party log management tools (like Splunk, Loggly, AlertLogic Log Manager, etc.) as long as the recommendation goal is achieved.
- The below Audit and Remediation steps need to be modified for your specific log management tool, as they are provided in the benchmark only for Amazon Cloudwatch

Rationale:

Separating log group destinations on a per tier basis allows unique settings to be applied on a per group basis for:

- Retention of logs
- Access Controls
- Export/Stream of data to other AWS Services for analysis/processing
    - AWS S3
    - AWS Lambda
    - AWS Elastic Search

Audit:

Using the Amazon unified command line interface:

- Search for your Web tier Cloudwatch log group:

```
aws logs describe-log-groups --query "logGroups[?logGroupName ==
'<web_tier_log_group>']"
```

Remediation:

Using the Amazon unified command line interface:

- Create a Cloudwatch log group for the Web tier:

```
aws logs create-log-group --log-group-name <web_tier_log_group>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/logs/describe-log-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/logs/create-log-group.html

5.5 Ensure Cloudwatch Log Group is created for App Tier (Scored)

Profile Applicability:

- Level 1

Description:

AWS CloudWatch Log groups define groups of log streams that share the same retention, monitoring, and access control settings. Each log stream has to belong to one log group.

Note:

- You can also use any third party log management tools (like Splunk, Loggly, AlertLogic Log Manager, etc.) as long as the recommendation goal is achieved.
- The below Audit and Remediation steps need to be modified for your specific log management tool, as they are provided in the benchmark only for Amazon Cloudwatch

Rationale:

Separating log group destinations on a per tier basis allows unique settings to be applied on a per group basis for:

- Retention of logs
- Access Controls
- Export/Stream of data to other AWS Services for analysis/processing
    - AWS S3
    - AWS Lambda
    - AWS Elastic Search

Audit:

Using the Amazon unified command line interface:

- Search for your App tier Cloudwatch log group:

```
aws logs describe-log-groups --query "logGroups[?logGroupName ==
'<app_tier_log_group>']"
```

Remediation:

Using the Amazon unified command line interface:

- Create a Cloudwatch log group for the App tier:

```
aws logs create-log-group --log-group-name <app_tier_log_group>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/logs/describe-log-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/logs/create-log-group.html

5.6 Ensure Cloudwatch Log Group for Web Tier has a retention period (Scored)

Profile Applicability:

- Level 1

Description:

Retention period should be used to specify how long log events are kept in CloudWatch Logs. Expired log events get deleted automatically. Just like metric filters, retention settings are also assigned to log groups, and the retention assigned to a log group is applied to their log streams.

Note:

- You can also use any third party log management tools (like Splunk, Loggly, AlertLogic Log Manager, etc.) as long as the recommendation goal is achieved.
- The below Audit and Remediation steps need to be modified for your specific log management tool, as they are provided in the benchmark only for Amazon Cloudwatch

Rationale:

Different log groups may require different retention periods, depending on operational and regulatory constraints.

Audit:

Using the Amazon unified command line interface:

- Search for your Web tier Cloudwatch log group, and check for the presence of "retentionInDays" element:

```
aws logs describe-log-groups --query "logGroups[?logGroupName ==
'<web_tier_log_group>']"
```

Remediation:

Using the Amazon unified command line interface:

- Put a retention policy for your Web tier Cloudwatch log group:

```
aws logs put-retention-policy --log-group-name <web_tier_log_group> --
retention-in-days <log_retention_period>
```

Impact:

If the retention period is not configured then logs will be retained indefinitely with increasing cost.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/logs/describe-log-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/logs/put-retention-policy.html

5.7 Ensure Cloudwatch Log Group for App Tier has a retention period (Scored)

Profile Applicability:

- Level 1

Description:

Retention period should be used to specify how long log events are kept in CloudWatch Logs. Expired log events get deleted automatically. Just like metric filters, retention settings are also assigned to log groups, and the retention assigned to a log group is applied to their log streams.

Note:

- You can also use any third party log management tools (like Splunk, Loggly, AlertLogic Log Manager, etc.) as long as the recommendation goal is achieved.
- The below Audit and Remediation steps need to be modified for your specific log management tool, as they are provided in the benchmark only for Amazon Cloudwatch

Rationale:

Different log groups may require different retention periods, depending on operational and regulatory constraints.

Audit:

Using the Amazon unified command line interface:

- Search for your App tier Cloudwatch log group, and check for the presence of "retentionInDays" element:

```
aws logs describe-log-groups --query "logGroups[?logGroupName ==
'<app_tier_log_group>']"
```

Remediation:

Using the Amazon unified command line interface:

- Put a retention policy for your App tier Cloudwatch log group:

```
aws logs put-retention-policy --log-group-name <app_tier_log_group> --
retention-in-days <log_retention_period>
```

Impact:

If the retention period is not configured then logs will be retained indefinitely with increasing cost.

References:

1. http://docs.aws.amazon.com/cli/latest/reference/logs/describe-log-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/logs/put-retention-policy.html

5.8 Ensure an agent for AWS Cloudwatch Logs is installed within Auto-Scaling Group for Web-Tier (Not Scored)

Profile Applicability:

- Level 1

Description:

You can use CloudWatch Logs to monitor, store and access log files from an Amazon EC2 instance (application or system data).

With CloudWatch Logs, you can monitor your logs, in near real-time, for specific phrases, values or patterns (metrics). For example, you could set an alarm on the number of errors that occur in your system logs or view graphs of web request latency from your application logs. Log data can be stored and accessed for as long as you need using highly durable, low-cost storage so you don't have to worry about filling up hard drives.

A Cloudwatch agent needs to run within the Guest Operating System of each EC2 instance you wish to ship logs from.

Note:

- You can also use any third party log management tools (like Splunk, Loggly, AlertLogic Log Manager, etc.) as long as the recommendation goal is achieved.
- The below Audit and Remediation steps need to be modified for your specific log management tool, as they are provided in the benchmark only for Amazon Cloudwatch

Rationale:

Allows for centralized logging, monitoring and incident reporting of both System level events and Application level events within EC2 instances.

Audit:

Using the Amazon unified command line interface:

- Check if the Cloudwatch Logs agent is installed through UserData in the Web tier Autoscaling Launch Configuration:

```
aws autoscaling describe-launch-configurations --launch-configuration-names
<web_tier_launch_config>--query "LaunchConfigurations[*].UserData"
```

- Output should be similar with:

```
#!/bin/bash
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-
setup.py -O
chmod +x ./awslogs-agent-setup.py
./awslogs-agent-setup.py -n -r us-east-1 -c
s3://<s3_bucket_name>/<cloudwatch_agent_config_file>
```

Remediation:

Using the Amazon unified command line interface:

- Create a sample agent configuration file for Amazon Linux and save it as a text file (for example, awslogs.cfg) either on the AMI's filesystem, in a publicly accessible http/https location, or an Amazon S3 location (for example, s3://<s3_bucket_name>/*<cloudwatch_agent_config_file>*):

```
[general]
state_file = /var/awslogs/state/agent-state

[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

- Create a new Web tier Autoscaling Launch Configuration with UserData populated for installing Cloudwatch Logs agent:
    - Create and save locally a file containing the UserData, for example /tmp/UserData.txt:

```
#!/bin/bashcurl https://s3.amazonaws.com/aws-
cloudwatch/downloads/latest/awslogs-agent-setup.py -Ochmod +x
./awslogs-agent-setup.py./awslogs-agent-setup.py -n -r us-east-1 -c
s3://<s3_bucket_name>/<cloudwatch_agent_config_file>
```

    - **Note:**
      You can install the CloudWatch Logs agent by specifying the us-east-1, us-west-1, us-west-2, eu-west-1, eu-central-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, or sa-east-1 regions.

```
aws autoscaling create-launch-configuration --launch-configuration-name
<web_tier_launch_config> --image-id <web_tier_ami> --key-name
<your_key_pair> --security-groups <web_tier_security_group> --instance-
type <desired_instance_type> --iam-instance-profile
<web_tier_instance_profile> --user-data file:///tmp/UserData.txt
```

References:

1. http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/EC2NewInstanceCWL.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html

5.9 Ensure an agent for AWS Cloudwatch Logs is installed within Auto-Scaling Group for App-Tier (Not Scored)

Profile Applicability:

- Level 1

Description:

You can use CloudWatch Logs to monitor, store and access log files from an Amazon EC2 instance (application or system data).

With CloudWatch Logs, you can monitor your logs, in near real-time, for specific phrases, values or patterns (metrics). For example, you could set an alarm on the number of errors that occur in your system logs or view graphs of web request latency from your application logs. Log data can be stored and accessed for as long as you need using highly durable, low-cost storage so you don't have to worry about filling up hard drives.

A Cloudwatch agent needs to run within the Guest Operating System of each EC2 instance you wish to ship logs from.

Note:

- You can also use any third party log management tools (like Splunk, Loggly, AlertLogic Log Manager, etc.) as long as the recommendation goal is achieved.
- The below Audit and Remediation steps need to be modified for your specific log management tool, as they are provided in the benchmark only for Amazon Cloudwatch

Rationale:

Allows for centralized logging, monitoring and incident reporting of both System level events and Application level events within EC2 instances.

Audit:

Using the Amazon unified command line interface:

- Check if the Cloudwatch Logs agent is installed through UserData in the App tier Autoscaling Launch Configuration:

```
aws autoscaling describe-launch-configurations --launch-configuration-names
<app_tier_launch_config> --query "LaunchConfigurations[*].UserData"
```

- Output should be similar with:

```
#!/bin/bash
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-agent-
setup.py -O
chmod +x ./awslogs-agent-setup.py
./awslogs-agent-setup.py -n -r us-east-1 -c
s3://<s3_bucket_name>/<cloudwatch_agent_config_file>
```

Remediation:

Using the Amazon unified command line interface:

- Create a sample agent configuration file for Amazon Linux and save it as a text file (for example, awslogs.cfg) either on the AMI's filesystem, in a publicly accessible http/https location, or an Amazon S3 location (for example, s3://<s3_bucket_name>/*<cloudwatch_agent_config_file>*):

```
[general]
state_file = /var/awslogs/state/agent-state

[/var/log/messages]
file = /var/log/messages
log_group_name = /var/log/messages
log_stream_name = {instance_id}
datetime_format = %b %d %H:%M:%S
```

- Create a new App tier Autoscaling Launch Configuration with UserData populated for installing Cloudwatch Logs agent:
    - Create and save locally a file containing the UserData, for example /tmp/UserData.txt:

```
#!/bin/bash
curl https://s3.amazonaws.com/aws-cloudwatch/downloads/latest/awslogs-
agent-setup.py -O
chmod +x ./awslogs-agent-setup.py
./awslogs-agent-setup.py -n -r us-east-1 -c
s3://<s3_bucket_name>/<cloudwatch_agent_config_file>
```

    - **Note:**
      You can install the CloudWatch Logs agent by specifying the us-east-1, us-west-1, us-west-2, eu-west-1, eu-central-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, or sa-east-1 regions.

```
aws autoscaling create-launch-configuration --launch-configuration-name
<app_tier_launch_config> --image-id <app_tier_ami> --key-name <your_key_pair>
--security-groups <app_tier_security_group> --instance-type
<desired_instance_type> --iam-instance-profile <span style="font-style:
normal;"><app_tier_instance_profile> --user-data file:///tmp/UserData.txt
```

References:

1. http://docs.aws.amazon.com/AmazonCloudWatch/latest/DeveloperGuide/EC2NewInstanceCWL.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html

5.10 Ensure an AWS Managed Config Rule for encrypted volumes is applied to Web Tier (Scored)

Profile Applicability:

- Level 1

Description:

AWS Config provides you with a detailed inventory of your AWS resources and their current configuration, and continuously records configuration changes to these resources.

You can evaluate these configurations and changes for compliance with ideal configurations as defined by AWS Config Rules.

Rationale:

Evaluation of Elastic Block Storage volume configuration to ensure encryption at rest is enabled which have been tagged as Web-Tier

Audit:

Using the Amazon unified command line interface:

- Search for a Config Rule that checks if the EBS volumes tagged with Web tier tags are encrypted with Web tier KMS key:

```
aws configservice describe-config-rules --query
"ConfigRules[?Source.SourceIdentifier == 'ENCRYPTED_VOLUMES']|[?Scope.TagKey
== '<web_tier_tag>']|[?Scope.TagValue ==
'<web_tier_tag_value>']|[?InputParameters ==
'{\"kmsId\":\"<web_tier_kms_key>\"}']"
```

Remediation:

Using the Amazon unified command line interface:

- Create locally a json file (similar with the below sample) with the configuration of the Config Rule, and save it as /tmp/ConfigRule.json:

```
{
  "Description": "Checks whether Web Tier EBS volumes that are in an attached
state are encrypted.",
  "ConfigRuleName": "encrypted-volumes-web-tier",
  "Source": {
    "Owner": "AWS",
    "SourceIdentifier": "ENCRYPTED_VOLUMES"
```

```
  },
  "InputParameters": "{\"kmsId\":\"<web_tier_kms_key>\"}",
  "Scope": {
    "TagKey": "<web_tier_tag>",
    "TagValue": "<web_tier_tag_value>"
  }
}
```

- Create a Config Rule using the configuration saved earlier:

```
aws configservice put-config-rule --config-rule file:///tmp/ConfigRule.json
```

References:

1. http://docs.aws.amazon.com/config/latest/developerguide/WhatIsConfig.html

5.11 Ensure an AWS Managed Config Rule for encrypted volumes is applied to App Tier (Scored)

Profile Applicability:

- Level 1

Description:

AWS Config provides you with a detailed inventory of your AWS resources and their current configuration, and continuously records configuration changes to these resources.

You can evaluate these configurations and changes for compliance with ideal configurations as defined by AWS Config Rules.

Rationale:

Evaluation of Elastic Block Storage volume configuration to ensure encryption at rest is enabled which have been tagged as App-Tier

Audit:

Using the Amazon unified command line interface:

- Search for a Config Rule that checks if the EBS volumes tagged with App tier tags are encrypted with App tier KMS key:

```
aws configservice describe-config-rules --query
"ConfigRules[?Source.SourceIdentifier == 'ENCRYPTED_VOLUMES']|[?Scope.TagKey
== '<app_tier_tag>']|[?Scope.TagValue ==
'<app_tier_tag_value>']|[?InputParameters ==
'{\"kmsId\":\"<app_tier_kms_key>\"}']"
```

Remediation:

Using the Amazon unified command line interface:

- Create locally a json file (similar with the below sample) with the configuration of the Config Rule, and save it as /tmp/ConfigRule.json:

```
{
  "Description": "Checks whether App Tier EBS volumes that are in an attached
state are encrypted.",
  "ConfigRuleName": "encrypted-volumes-app-tier",
  "Source": {
    "Owner": "AWS",
    "SourceIdentifier": "ENCRYPTED_VOLUMES"
```

```
  },
  "InputParameters": "{\"kmsId\":\"<app_tier_kms_key>\"}",
  "Scope": {
    "TagKey": "<app_tier_tag>",
    "TagValue": "<app_tier_tag_value>"
  }
}
```

- Create a Config Rule using the configuration saved earlier:

```
aws configservice put-config-rule --config-rule file:///tmp/ConfigRule.json
```

References:

1. http://docs.aws.amazon.com/config/latest/developerguide/WhatIsConfig.html
2. http://docs.aws.amazon.com/cli/latest/reference/configservice/describe-config-rules.html
3. http://docs.aws.amazon.com/cli/latest/reference/configservice/put-config-rule.html

5.12 Ensure an AWS Managed Config Rule for EIPs attached to EC2 instances within VPC (Scored)

Profile Applicability:

- Level 1

Description:

AWS Config provides you with a detailed inventory of your AWS resources and their current configuration, and continuously records configuration changes to these resources.

You can evaluate these configurations and changes for compliance with ideal configurations as defined by AWS Config Rules.

Rationale:

Evaluation of EC2 instance configuration to ensure there are no publicly addressable IP's attached which would violate the defence in depth model

Audit:

Using the Amazon unified command line interface:

- Search for a Config Rule that checks whether all EIP addresses allocated to a VPC are attached to EC2 instances or in-use ENIs:

```
aws configservice describe-config-rules --query
"ConfigRules[?Source.SourceIdentifier == 'EIP_ATTACHED']"
```

Remediation:

Using the Amazon unified command line interface:

- Create locally a json file (similar with the below sample) with the configuration of the Config Rule, and save it as /tmp/ConfigRule.json:

```
"Description": "Checks whether all EIP addresses allocated to a VPC are
attached to EC2 instances or in-use ENIs.",
"ConfigRuleName": "eip-attached",
"Source": {
  "Owner": "AWS",
  "SourceIdentifier": "EIP_ATTACHED"
},
"Scope": {
  "ComplianceResourceTypes": [
    "AWS::EC2::EIP"
```

```
      ]
    }
}
```

- Create a Config Rule using the configuration saved earlier:

```
aws configservice put-config-rule --config-rule file:///tmp/ConfigRule.json
```

References:

1. http://docs.aws.amazon.com/config/latest/developerguide/WhatIsConfig.html
2. http://docs.aws.amazon.com/cli/latest/reference/configservice/describe-config-rules.html
3. http://docs.aws.amazon.com/cli/latest/reference/configservice/put-config-rule.html

6 Networking

This section provides recommendations on securing the networking aspects of the application stack.

6.1 Ensure Root Domain Alias Record Points to ELB (Scored)

Profile Applicability:

- Level 2

Description:

Amazon Route 53 translates friendly domains names like www.example.com into IP addresses like 192.0.2.1. Amazon Route 53 responds to DNS queries using a global network of authoritative DNS servers, which reduces latency.

When someone enters your domain name in a browser, a DNS request is forwarded to the nearest Amazon Route 53 DNS server in a global network of authoritative DNS servers. Amazon Route 53 responds with the IP address that you specified.

Each domain has an associated hosted zone which contains the resource records pointing to each layer of the application.

A private hosted zone is a container that holds information about how you want to route traffic for a domain and its subdomains within the Amazon Virtual Private Cloud (Amazon VPC). To begin, you create a private hosted zone and specify the Amazon VPCs that you want to associate with the hosted zone. You then create resource record sets that determine how Amazon Route 53 responds to queries for your domain and subdomains within and among your Amazon VPCs.

Rationale:

Route53 provides special record type called Alias that allow to create an A record for the root domain and point it to the fully qualified domain of the Elastic Load Balancer (ELB) associated with the web-server layer or Amazon CloudFront.

In the same way records for all other layers should be created in order to allow flexibility in the application design and not hard-code the FQDN of a resource.

Audit:

Using the Amazon unified command line interface:

- List all the hosted zones and check if the domain name used by the application is present among them:

```
aws route53 list-hosted-zones --query 'HostedZones[*].{Name:Name, Id:Id}' --
output table
```

Remediation:

Using the Amazon unified command line interface:

- Create a hosted zone for YourDomain.com:

```
aws route53 create-hosted-zone --name <your_domain.com> --caller-reference
<any_string>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/route53/list-hosted-zones.html
2. http://docs.aws.amazon.com/cli/latest/reference/route53/create-hosted-zone.html
3. http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/AboutHostedZones.html

6.2 Ensure a DNS alias record for the root domain (Scored)

Profile Applicability:

- Level 2

Description:

While ordinary Amazon Route 53 resource record sets are standard DNS resource record sets, *alias resource record sets* provide an Amazon Route 53–specific extension to DNS functionality. Instead of an IP address or a domain name, an alias resource record set contains a pointer to a CloudFront distribution or an ELB load balancer.

Alias resource record sets can save you time because Amazon Route 53 automatically recognizes changes in the resource record sets that the alias resource record set refers to.

Rationale:

In order to point the root domain to a CloudFront distribution or an Elastic Load Balancer (ELB), an alias resource record set should be created.

Audit:

Using the Amazon unified command line interface:

- List only the Alias records of the hosted zone used by the application. Use the hosted zone id from the previous recommendation. Check if an Alias record is created for the root domain:

```
aws route53 list-resource-record-sets --hosted-zone-id  <your_hosted_zone_id>
--query 'ResourceRecordSets[?AliasTarget != null]'
```

Remediation:

First create a json file that represents the alias record that you want to add, and save it locally as "alias.json". Below you can find a simple alias record representation:

```
{
  "Changes": [
    {
      "Action": "CREATE",
      "ResourceRecordSet": {
        "Name": "<your_root_domain>",
        "Type": "A",
        "AliasTarget": {
          "HostedZoneId": "hosted zone ID for your CloudFront distribution,
Amazon S3 bucket, Elastic Load Balancing load balancer, or Amazon Route 53
```

```
hosted zone",
          "DNSName": "DNS domain name for your CloudFront distribution,
Amazon S3 bucket, Elastic Load Balancing load balancer, or another resource
record set in this hosted zone",
          "EvaluateTargetHealth": false
        }
      }
    }
  ]
}
```

Using the Amazon unified command line interface:

- Create an Alias records in your hosted zone:

```
aws route53 change-resource-record-sets --hosted-zone-id
<your_hosted_zone_id> --change-batch file:///PathTo/alias.json
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/route53/list-resource-record-sets.html
2. http://docs.aws.amazon.com/cli/latest/reference/route53/change-resource-record-sets.html
3. http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-choosing-alias-non-alias.html

6.3 Use CloudFront Content Distribution Network (Scored)

Profile Applicability:

- Level 1

Description:

Amazon CloudFront can be used to deliver either the entire website, including dynamic, static, streaming, and interactive content using a global network of edge locations. Requests for your content are automatically routed to the nearest edge location, so content is delivered with the best possible performance. Amazon CloudFront is optimized to work with other Amazon Web Services, like Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Load Balancing, and Amazon Route 53.

Amazon CloudFront gives you three options for accelerating your entire website while delivering your content securely over HTTPS from all of CloudFront's edge locations. In addition to delivering securely from the edge, you can also configure CloudFront to use HTTPS connections for origin fetches so that your data is encrypted end-to-end from your origin to your end users.

Rationale:

Using the Amazon CloudFront content delivery network will provide improved performance with the application as the content is delivered from the closest edge location in terms of DNS resolution latency.

It improves the ability of the application to absorb and mitigate potential distributed denial of service (DDOS) attacks and keep the application available for legitimate users.

The content can be delivered securely over HTTPS from all edge location between the customer and between the edge locations and the origin.

Audit:

Using the Amazon unified command line interface:

- List the Cloudfront distributions present in the AWS account, and check in the aliases field for the presence of the domain name used by the application:

```
aws cloudfront list-distributions --query "DistributionList.Items[*].{Id:Id,
Status:Status, DomainName:DomainName, Aliases:Aliases.Items}"
```

Remediation:

Using the Amazon unified command line interface:

- You can either create a Cloudfront distribution only by specifying the origin domain name (ELB, S3 bucket or web server):

```
aws cloudfront create-distribution --origin-domain-name
<your_original_domain_name> --default-root-object index.html
```

- Or by creating locally a distribution config file distconfig.json with all the Cloudfront distribution parameters:

```
aws cloudfront create-distribution --distribution-config
file://distconfig.json
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/list-distributions.html
2. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/create-distribution.html
3. http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/distribution-working-with.html

6.4 Ensure Geo-Restriction is enabled within Cloudfront Distribution (Scored)

Profile Applicability:

- Level 2

Description:

Amazon CloudFront is a web service that speeds up distribution of your static and dynamic web content, for example, .html, .css, .php, image, and media files, to end users. CloudFront delivers your content through a worldwide network of edge locations. When an end user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency, so content is delivered with the best possible performance. If the content is already in that edge location, CloudFront delivers it immediately. If the content is not currently in that edge location, CloudFront retrieves it from a customer defined Origin, such as AWS S3, AWS ELB or EC2.

Rationale:

Provides the ability to block IP addresses based on Geo IP from reaching your CDN or Web Application resources. Can be used to assist in mitigation of DoS attacks.

Audit:

Using the Amazon unified command line interface:

- Check if geo restrictions are enabled on your application CloudFront distribution:

```
aws cloudfront get-distribution --id <application_cfn_distribution_id> --
query "Distribution.DistributionConfig.Restrictions.GeoRestriction"
```

Remediation:

Using the Amazon unified command line interface:

- For enabling GeoRestrictions first save locally the current distribution config:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "DistributionConfig" > /tmp/cf-distribution.json
```

- Edit the GeoRestrictions section in /tmp/cf-distribution.json with the desired configuration (similar to the below sample):

```
"Restrictions": {
  "GeoRestriction": {
    "RestrictionType": "`<blacklist|whitelist>`",
    "Quantity": 3,
    "Items": ["`<country_code_1>`", "`<country_code_2>`"]
  }
},
```

- Retrieve the current ETag of your CloudFront distribution:

```
aws cloudfront get-distribution-config --id <application_cfn_distribution_id>
--query "ETag"
```

- Update the CloudFront distribution using the edited config and the above Etag:

```
aws cloudfront update-distribution --id <application_cfn_distribution_id> --
distribution-config file:///tmp/cf-distribution.json --if-match
<application_cfn_distribution_etag>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/get-distribution.html
2. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/get-distribution.html
3. http://docs.aws.amazon.com/cli/latest/reference/cloudfront/update-distribution.html

6.5 Ensure subnets for the Web tier ELB are created (Scored)

Profile Applicability:

- Level 1

Description:

You can create a VPC that spans multiple Availability Zones. After creating a VPC, you can add one or more subnets in each Availability Zone. Each subnet must reside entirely within one Availability Zone and cannot span zones. Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. AWS assigns a unique ID to each subnet.

When you create a subnet, you specify the CIDR block for the subnet. The CIDR block of a subnet shouldn't be the same as the CIDR block for the VPC (for a single subnet in the VPC). The allowed block size is between a /28 netmask and /16 netmask. If you create more than one subnet in a VPC, the CIDR blocks of the subnets must not overlap.

Some AWS regions have more than 2 availability zones and it is recommended to use more than 2 where possible.

Rationale:

At least 2 subnets in 2 different availability zones (AZ) should be created in order to have fault tolerance and high availability from the perspective of resource deployment.

Audit:

Using the Amazon unified command line interface:

- List the subnets associated with the Web tier ELB:

```
aws elb describe-load-balancers --load-balancer-name <web_tier_elb> --query
"LoadBalancerDescriptions[*].Subnets"
```

OR
Using the Amazon unified command line interface:

- List the subnets created for the Web tier ELB:

```
aws ec2 describe-subnets --filters
Name=tag:<public_tier_tag>,Values=<public_tier_tag_value> --query
"Subnets[*].SubnetId"
```

Remediation:

Using the Amazon unified command line interface:

- Create subnets for Web tier ELB, and note the subnet id:

```
aws ec2 create-subnet --vpc-id <application_vpc> --cidr-block <desired_cidr>
```

- Tag the above subnets with the Web tier ELB tags:

```
aws ec2 create-tags --resources <web_tier_elb_subnet1> <web_tier_elb_subnet2>
--tags Key=<public_tier_tag>,Value=<public_tier_tag_value>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-tags.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-subnet.html

6.6 Ensure subnets for the Web tier are created (Scored)

Profile Applicability:

- Level 1

Description:

You can create a VPC that spans multiple Availability Zones. After creating a VPC, you can add one or more subnets in each Availability Zone. Each subnet must reside entirely within one Availability Zone and cannot span zones. Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. AWS assigns a unique ID to each subnet.

When you create a subnet, you specify the CIDR block for the subnet. The CIDR block of a subnet shouldn't be the same as the CIDR block for the VPC (for a single subnet in the VPC). The allowed block size is between a /28 netmask and /16 netmask. If you create more than one subnet in a VPC, the CIDR blocks of the subnets must not overlap.

Some AWS regions have more than 2 availability zones and it is recommended to use more than 2 where possible.

Rationale:

At least 2 subnets in 2 different availability zones (AZ) should be created in order to have fault tolerance and high availability from the perspective of resource deployment.

Audit:

Using the Amazon unified command line interface:

- List the subnets associated with the Web tier:

```
aws ec2 describe-subnets --filters
Name=tag:<web_tier_tag>,Values=<web_tier_tag_value> --query
"Subnets[*].SubnetId"
```

Remediation:

Using the Amazon unified command line interface:

- Create subnets for Web tier, and note the subnet id:

```
aws ec2 create-subnet --vpc-id <application_vpc> --cidr-block <desired_cidr>
```

- Tag the above subnets with the Web tier tags:

```
aws ec2 create-tags --resources <web_tier_subnet1> <web_tier_subnet2> --tags
Key=<web_tier_tag>,Value=<web_tier_tag_value>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-tags.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-subnet.html

6.7 Ensure subnets for the App tier are created (Scored)

Profile Applicability:

- Level 1

Description:

You can create a VPC that spans multiple Availability Zones. After creating a VPC, you can add one or more subnets in each Availability Zone. Each subnet must reside entirely within one Availability Zone and cannot span zones. Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. AWS assigns a unique ID to each subnet.

When you create a subnet, you specify the CIDR block for the subnet. The CIDR block of a subnet shouldn't be the same as the CIDR block for the VPC (for a single subnet in the VPC). The allowed block size is between a /28 netmask and /16 netmask. If you create more than one subnet in a VPC, the CIDR blocks of the subnets must not overlap.

Some AWS regions have more than 2 availability zones and it is recommended to use more than 2 where possible.

Rationale:

At least 2 subnets in 2 different availability zones (AZ) should be created in order to have fault tolerance and high availability from the perspective of resource deployment.

Audit:

Using the Amazon unified command line interface:

- List the subnets associated with the App tier:

```
aws ec2 describe-subnets --filters
Name=tag:<app_tier_tag>,Values=<app_tier_tag_value> --query
"Subnets[*].SubnetId"
```

Remediation:

Using the Amazon unified command line interface:

- Create subnets for App tier, and note the subnet id:

```
aws ec2 create-subnet --vpc-id <application_vpc> --cidr-block <desired_cidr>
```

- Tag the above subnets with the App tier tags:

```
aws ec2 create-tags --resources <app_tier_subnet1> <app_tier_subnet2> --tags
Key=<app_tier_tag>,Value=<app_tier_tag_value>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-tags.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-subnet.html

6.8 Ensure subnets for the Data tier are created (Scored)

Profile Applicability:

- Level 1

Description:

You can create a VPC that spans multiple Availability Zones. After creating a VPC, you can add one or more subnets in each Availability Zone. Each subnet must reside entirely within one Availability Zone and cannot span zones. Availability Zones are distinct locations that are engineered to be isolated from failures in other Availability Zones. By launching instances in separate Availability Zones, you can protect your applications from the failure of a single location. AWS assigns a unique ID to each subnet.

When you create a subnet, you specify the CIDR block for the subnet. The CIDR block of a subnet shouldn't be the same as the CIDR block for the VPC (for a single subnet in the VPC). The allowed block size is between a /28 netmask and /16 netmask. If you create more than one subnet in a VPC, the CIDR blocks of the subnets must not overlap.

Some AWS regions have more than 2 availability zones and it is recommended to use more than 2 where possible.

Rationale:

At least 2 subnets in 2 different availability zones (AZ) should be created in order to have fault tolerance and high availability from the perspective of resource deployment.

Audit:

Using the Amazon unified command line interface:

- List the subnets associated with the Data tier:

```
aws ec2 describe-subnets --filters
Name=tag:<data_tier_tag>,Values=<data_tier_tag_value> --query
"Subnets[*].SubnetId"
```

Remediation:

Using the Amazon unified command line interface:

- Create subnets for Data tier, and note the subnet id:

```
aws ec2 create-subnet --vpc-id <application_vpc> --cidr-block <desired_cidr>
```

- Tag the above subnets with the Data tier tags:

```
aws ec2 create-tags --resources SubnetId1 SubnetId2 --tags
Key=<data_tier_tag>,Value=<data_tier_tag_value>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-tags.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-subnet.html

6.9 Ensure Elastic IPs for the NAT Gateways are allocated (Scored)

Profile Applicability:

- Level 1

Description:

An Elastic IP address is a static, public IP address designed for dynamic cloud computing. You can associate an Elastic IP address with any instance, network interface for your VPC or a NAT Gateway. With an Elastic IP address, you can mask the failure of an instance by rapidly remapping the address to another instance in your VPC.

Rationale:

In order to be able to create NAT Gateways that allow Internet access from the private subnet of the VPC, Elastic IPs should be allocated for each NAT Gateway.

Some AWS Regions have more than 2 Availability Zones, in this case it is recommended to allocate an Elastic IP to each NAT Gateway in each of the public subnets used.

Audit:

Using the Amazon unified command line interface:

- Check if you have Elastic IP addresses allocated and unused for the number of NAT Gateways that you want to deploy:

```
aws ec2 describe-addresses --filters Name=domain,Values=vpc --query
"Addresses[?AssociationId == null]"
```

Remediation:

Using the Amazon unified command line interface:

- Allocate Elastic IP addreses for the number of NAT Gateways that you want to deploy:

```
aws ec2 allocate-address --domain vpc
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/allocate-address.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-addresses.html

6.10 Ensure NAT Gateways are created in at least 2 Availability Zones (Scored)

Profile Applicability:

- Level 1

Description:

You can use a network address translation (NAT) gateway to enable instances in a private subnet to connect to the Internet or other AWS services, but prevent the Internet from initiating a connection with those instances.

To create a NAT gateway, you must specify the public subnet in which the NAT gateway will reside. You must also specify an Elastic IP address to associate with the NAT gateway when you create it. This enables instances in your private subnets to communicate with the Internet.

Each NAT gateway is created in a specific Availability Zone and implemented with redundancy in that zone

Rationale:

In order to enable instances in a private subnets to connect to the Internet or other AWS services, but prevent the Internet from initiating a connection with those instances, NAT Gateways should be created in at least 2 Availability Zones.

Some AWS Regions have more than 2 Availability Zones, in this case it is recommended to create a NAT Gateway in each of the public subnets used.

Audit:

Using the Amazon unified command line interface:

- List the NAT Gateways from your application VPC, and note the subnets they are deployed in:

```
aws ec2 describe-nat-gateways --filter Name=vpc-id,Values=<application_vpc> --query "NatGateways[*].{NatGatewayId:NatGatewayId, SubnetId:SubnetId}"
```

- Check the Availability Zones where the above subnets are deployed:

```
aws ec2 describe-subnets --subnet-ids <public_subnet1> <public_subnet2> --query "Subnets[*].{SubnetId:SubnetId, AvailabilityZone:AvailabilityZone}"
```

Remediation:

Using the Amazon unified command line interface:

- Create a NAT Gateway in a public subnet from a different Availability Zone:

```
aws ec2 create-nat-gateway --subnet-id <public_subnet1> --allocation-id
<elastic_ip_allocation>
```

and/or

```
aws ec2 create-nat-gateway --subnet-id <public_subnet2> --allocation-id
<elastic_ip_allocation>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-nat-gateways.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-nat-gateway.html

6.11 Ensure a route table for the public subnets is created (Scored)

Profile Applicability:

- Level 1

Description:

A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed.

Each subnet in your VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table.

Rationale:

Once a route table for the public subnet is created, all the subnets which should be public in the Web ELB tier can be associated with the public subnet.

The private subnet should only contain the default route (0.0.0.0/0) pointing to the Internet Gateway (IGW).

Audit:

Using the Amazon unified command line interface:

- List route tables attached to the public subnets, and check if they contain the default route (0.0.0.0/0) pointing to the Internet Gateway (IGW):

```
aws ec2 describe-route-tables --filters Name=association.subnet-
id,Values=<public_subnet1>,<public_subnet2> --query
"RouteTables[*].{RouteTableId:RouteTableId, Tags:Tags, Routes:Routes}"
```

Remediation:

Using the Amazon unified command line interface:

- Create a route table for your public subnets, and note the RouteTableId in the output:

```
aws ec2 create-route-table --vpc-id <application_vpc>
```

- Associate the new route table with the public subnets:

```
aws ec2 associate-route-table --route-table-id <route_table_id> --subnet-id
<public_subnet1>
aws ec2 associate-route-table --route-table-id <route_table_id> --subnet-id
<public_subnet2>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-route-tables.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-route-table.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-route.html

6.12 Ensure a route table for the private subnets is created (Scored)

Profile Applicability:

- Level 1

Description:

A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed.

Each subnet in your VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table.

Rationale:

Once a route table for the private subnet is created, all the subnets which should be private in the Web, App and Data tiers can be associated with the route table.

The route table should only contain the default route (0.0.0.0/0) pointing to the NAT Gateway.

Audit:

Using the Amazon unified command line interface:

- List route tables attached to the private subnets, and check if they contain the default route (0.0.0.0/0) pointing to the NAT Gateway:

```
aws ec2 describe-route-tables --filters Name=association.subnet-
id,Values=<private_subnet1>,<private_subnet2> --query
"RouteTables[*].{RouteTableId:RouteTableId, Tags:Tags, Routes:Routes}"
```

Remediation:

Using the Amazon unified command line interface:

- Create a route table for your private subnets, and note the RouteTableId in the output:
    - aws ec2 create-route-table --vpc-id <application_vpc>
- Associate the new route table with the private subnets:

```
aws ec2 associate-route-table --route-table-id <route_table_id> --subnet-id
<private_subnet1>
```

```
aws ec2 associate-route-table --route-table-id <route_table_id> --subnet-id
<private_subnet2>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-route-tables.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-route-table.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-route.html

6.13 Ensure Routing Table associated with Web tier ELB subnet have the default route (0.0.0.0/0) defined to allow connectivity to the VPC Internet Gateway (IGW) (Scored)

Profile Applicability:

- Level 1

Description:

A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed.

Each subnet in your VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table.

Rationale:

The default route (0.0.0.0/0) should be pointing to the Internet Gateway in order to provide internet connectivity for the Web tier ELB.

Audit:

Using the Amazon unified command line interface:

- List the subnets associated with the Web tier ELB:

```
aws elb describe-load-balancers --load-balancer-name <web_tier_elb> --query
"LoadBalancerDescriptions[*].Subnets"
```

- List the routes of the route tables associated with the above subnets, and check if the default route (0.0.0.0/0) has an IGW configured as gateway:

```
aws ec2 describe-route-tables --filters Name=association.subnet-
id,Values=<web_tier_elb_subnet1>,<web_tier_elb_subnet2> --query
"RouteTables[*].{RouteTableId:RouteTableId, Routes:Routes}"
```

Remediation:

Using the Amazon unified command line interface:

- For the above route tables, if the default route (0.0.0.0/0) exists but it doesn't have an IGW configured as gateway:

```
aws ec2 replace-route --route-table-id <route_table_id> --destination-cidr-
block 0.0.0.0/0 --gateway-id <vpc_igw>
```

- For the above route tables, if the default route (0.0.0.0/0) doesn't exist:

```
aws ec2 create-route --route-table-id <route_table_id> --destination-cidr-
block 0.0.0.0/0 --gateway-id <vpc_igw>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/replace-route.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-route.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-route-tables.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html

6.14 Ensure Routing Table associated with Web tier subnet have the default route (0.0.0.0/0) defined to allow connectivity to the VPC NAT Gateway (Scored)

Profile Applicability:

- Level 1

Description:

A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed.

Each subnet in your VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table.

Rationale:

The default route (0.0.0.0/0) should be pointing to the NAT Gateway in order to provide internet connectivity for the Web tier instances.

Audit:

Using the Amazon unified command line interface::

- List the subnets associated with the Web tier:

```
aws ec2 describe-subnets --filters
Name=tag:<web_tier_tag>,Values=<web_tier_tag_value> --query
"Subnets[*].SubnetId"
```

- List the routes of the route tables associated with the above subnets, and check if the default route (0.0.0.0/0) has a NAT GW configured as gateway:

```
aws ec2 describe-route-tables --filters Name=association.subnet-
id,Values=<web_tier_subnet1>,<web_tier_subnet2> --query
"RouteTables[*].{RouteTableId:RouteTableId, Routes:Routes}"
```

Remediation:

Using the Amazon unified command line interface:

- For the above route tables, if the default route (0.0.0.0/0) exists but it doesn't have a NAT GW configured as gateway:

```
aws ec2 replace-route --route-table-id <route_table_id> --destination-cidr-
block 0.0.0.0/0 --gateway-id <vpc_nat_gw>
```

- For the above route tables, if the default route (0.0.0.0/0) doesn't exist:

```
aws ec2 create-route --route-table-id <route_table_id> --destination-cidr-
block 0.0.0.0/0 --gateway-id <vpc_nat_gw>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-route.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/replace-route.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-route-tables.html

6.15 Ensure Routing Table associated with App tier subnet have the default route (0.0.0.0/0) defined to allow connectivity to the VPC NAT Gateway (Scored)

Profile Applicability:

- Level 1

Description:

A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed.

Each subnet in your VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table.

Rationale:

The default route (0.0.0.0/0) should be pointing to the NAT Gateway in order to provide internet connectivity for the App tier instances.

Audit:

Using the Amazon unified command line interface::

- List the subnets associated with the App tier:

```
aws ec2 describe-subnets --filters
Name=tag:<app_tier_tag>,Values=<app_tier_tag_value> --query
"Subnets[*].SubnetId"
```

- List the routes of the route tables associated with the above subnets, and check if the default route (0.0.0.0/0) has a NAT GW configured as gateway:

```
aws ec2 describe-route-tables --filters Name=association.subnet-
id,Values=<app_tier_subnet1>,<app_tier_subnet2> --query
"RouteTables[*].{RouteTableId:RouteTableId, Routes:Routes}"
```

Remediation:

Using the Amazon unified command line interface:

- For the above route tables, if the default route (0.0.0.0/0) exists but it doesn't have a NAT GW configured as gateway:

    - aws ec2 replace-route --route-table-id *<route_table_id>* --destination-cidr-block 0.0.0.0/0 --gateway-id *<vpc_nat_gw>*
- For the above route tables, if the default route (0.0.0.0/0) doesn't exist:
    - aws ec2 create-route --route-table-id *<route_table_id>* --destination-cidr-block 0.0.0.0/0 --gateway-id *<vpc_nat_gw>*

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-route-tables.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/replace-route.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-route.html

6.16 Ensure Routing Table associated with Data tier subnet have NO default route (0.0.0.0/0) defined to allow connectivity to the VPC NAT Gateway (Scored)

Profile Applicability:

- Level 1

Description:

A *route table* contains a set of rules, called *routes*, that are used to determine where network traffic is directed.

Each subnet in your VPC must be associated with a route table; the table controls the routing for the subnet. A subnet can only be associated with one route table at a time, but you can associate multiple subnets with the same route table.

Rationale:

The default route (0.0.0.0/0) should not exist pointing to the NAT Gateway in order to restrict internet connectivity for the Data tier instances.

Audit:

Using the Amazon unified command line interface:

- List the subnets associated with the Data tier:

```
aws ec2 describe-subnets --filters
Name=tag:<data_tier_tag>,Values=<data_tier_tag_value> --query
"Subnets[*].SubnetId"
```

- List the routes of the route tables associated with the above subnets, and check if the default route (0.0.0.0/0) has a NAT GW configured as gateway:

```
aws ec2 describe-route-tables --filters Name=association.subnet-
id,Values=<data_tier_subnet1>,<data_tier_subnet2> --query
"RouteTables[*].{RouteTableId:RouteTableId, Routes:Routes}"
```

Remediation:

Using the Amazon unified command line interface:

- For the above route tables, if the default route (0.0.0.0/0) exists and it has a NAT GW configured as gateway:

- aws ec2 delete-route --route-table-id <*route_table_id*> --destination-cidr-block 0.0.0.0/0

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-route-tables.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/delete-route.html

6.17 Use a Web-Tier ELB Security Group to accept only HTTP/HTTPS (Scored)

Profile Applicability:

- Level 1

Description:

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in the AWS Virtual Private Cloud (VPC), you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

Rationale:

The SG associated with the Web tier ELB should allow connectivity from any source IP (0.0.0.0/0) only for the HTTP (TCP 80) and HTTPS (TCP 443) ports.

Audit:

Using the Amazon unified command line interface:

- List the security group associated with the Web tier ELB:

```
aws elb describe-load-balancers --load-balancer-name <your_web_tier_elb> --
query "LoadBalancerDescriptions[*].{DNSName:DNSName,
SecurityGroups:SecurityGroups, SourceSecurityGroup:SourceSecurityGroup}" --
output table
```

- note the security group id from the output
- List the ingress rules for the above security group, and make sure that only HTTP (TCP 80) and HTTPS (TCP 443) traffic is allowed from any source IP (0.0.0.0/0)any source IP (0.0.0.0/0):

```
aws ec2 describe-security-groups --group-ids  <security_group_id> --query
"SecurityGroups[*].{GroupName:GroupName, IpPermissions:IpPermissions}" --
output table
```

Remediation:

Using the Amazon unified command line interface:

- First remove all the ingress rules for the security group associated with the Web tier ELB:

```
aws ec2 describe-security-groups --group-id <security_group_id> --query
"SecurityGroups[0].IpPermissions" > /tmp/IpPermissions.json
aws ec2 revoke-security-group-ingress --group-id <security_group_id> --ip-
permissions file:///tmp/IpPermissions.json
```

- create locally the below json file containing ingress rules for any source IP (0.0.0.0/0) only for the HTTP (TCP 80) and HTTPS (TCP 443) ports and name it IpPermissions.json:

```
[
    {
        "PrefixListIds": [],
        "FromPort": 80,
        "IpRanges": [
            {
                "CidrIp": "0.0.0.0/0"
            }
        ],
        "ToPort": 80,
        "IpProtocol": "tcp",
        "UserIdGroupPairs": []
    },
    {
        "PrefixListIds": [],
        "FromPort": 443,
        "IpRanges": [
            {
                "CidrIp": "0.0.0.0/0"
            }
        ],
        "ToPort": 443,
        "IpProtocol": "tcp",
        "UserIdGroupPairs": []
    }
]
```

- Add to the security group associated with the Web tier ELB the above ingress rules:

```
aws ec2 authorize-security-group-ingress --group-id <security_group_id>  --
ip-permissions file:///PathTo/IpPermissions.json
```

6.18 Ensure Web tier ELB Security Group is not used in the Auto Scaling launch configuration of any other tier (Web, App) (Scored)

Profile Applicability:

- Level 1

Description:

When you use the AWS Management Console to create a load balancer in a VPC, you can choose an existing security group for the VPC or create a new security group for the VPC. If you choose an existing security group, it must allow traffic in both directions to the listener and health check ports for the load balancer. If you choose to create a security group, the console automatically adds rules to allow all traffic on these ports.

Be sure to review the security group rules to ensure that they allow traffic on the listener and health check ports for the new load balancer. When you delete your load balancer, this security group is not deleted automatically.

If you add a listener to an existing load balancer, you must review your security groups to ensure they allow traffic on the new listener port in both directions.

Rationale:

The web-tier ELB is the only one public facing and should have rules to allow inbound traffic the application ports (ex: HTTP and HTTPS) from any IP source (0.0.0.0/0).

The outbound security group rules for the web-tier ELB should be restricted to only the backend web-server instances for the appropriate application ports.

Associating the web-tier ELB security group to any other instances that shouldn't be publicly accessible exposes them to unauthorized access.

Audit:

Using the Amazon unified command line interface:

- List the security group associated with the Web tier ELB:

```
aws elb describe-load-balancers --load-balancer-name <web_tier_elb> --query
"LoadBalancerDescriptions[*].{DNSName:DNSName,
WebTierELBSecurityGroups:SecurityGroups,
SourceSecurityGroup:SourceSecurityGroup}" --output table
```

- note the "WebTierELBSecurityGroups" from the output

- List the security groups associated with all Autoscaling launch configurations and make sure that the above Web tier ELB security group is not present:

```
aws autoscaling describe-launch-configurations --query
"LaunchConfigurations[*].{SecurityGroups:SecurityGroups,
LaunchConfigurationName:LaunchConfigurationName}"
```

Remediation:

Using the Amazon unified command line interface:

- Create new launch configuration using the correct security groups for Web and/or App tier:

```
aws autoscaling create-launch-configuration --launch-configuration-name
<web_tier_launch_config> --image-id <web_tier_ami> --key-name <your_key_pair>
--security-groups <web_tier_security_group>/<app_tier_security_group> --
instance-type <desired_instance_type> --iam-instance-profile
<web_tier_instance_profile>/<app_tier_instance_profile>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html

6.19 Create the Web tier Security Group and ensure it allows inbound connections from Web tier ELB Security Group for explicit ports (Scored)

Profile Applicability:

- Level 1

Description:

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in the AWS Virtual Private Cloud (VPC), you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

This is required for both the configured port and protocol for the listener on the back-end instance and the port and protocol used for the health check.

Rationale:

This protects the Web-server tier from unauthorized access, it is recommended to add inbound security group rules that allow traffic for the specific application protocol and ports by referencing as source the security group associated with the Web tier ELB.

Audit:

Using the Amazon unified command line interface:

- List the ingress rules for the Web tier security group, and make sure that allows connections only from Web tier ELB security group on specific ports:

```
aws ec2 describe-security-groups --filters
Name=tag:<web_tier_tag>,Values=<web_tier_tag_value> --query
"SecurityGroups[*].{GroupName:GroupName, IpPermissions:IpPermissions,
WebTierSecurityGroup:GroupId}" --output table
```

Remediation:

Using the Amazon unified command line interface:

- First remove all the ingress rules for the Web tier security group (use the "WebTierSecurityGroup" element from Audit procedure):

```
aws ec2 describe-security-groups --group-id <web_tier_security_group> --query
"SecurityGroups[0].IpPermissions" > /tmp/IpPermissions.json
aws ec2 revoke-security-group-ingress --group-id <web_tier_security_group> --
ip-permissions file:///tmp/IpPermissions.json
```

- Add an ingress rule for a specific port, using --source-group option to specify the Web tier ELB security group as the source of the connections:

```
aws ec2 authorize-security-group-ingress --group-id <web_tier_security_group>
--protocol tcp --port <specific_port> --source-group
<web_tier_elb_security_group>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/authorize-security-group-ingress.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/revoke-security-group-ingress.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-security-groups.html

6.20 Ensure Web tier Security Group has no inbound rules for CIDR of 0 (Global Allow) (Scored)

Profile Applicability:

- Level 1

Description:

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in the AWS Virtual Private Cloud (VPC), you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

Rationale:

Considering any of the non-public tiers receive requests only either from the upper tier or from resources inside the same VPC, any inbound rules that allow traffic from any source (0.0.0.0/0) are not necessary and should be removed.

Audit:

Using the Amazon unified command line interface:

- List the ingress rules for the Web tier security group, and make sure it has no inbound rules for CIDR of 0.0.0.0/0:

```
aws ec2 describe-security-groups --filters
Name=tag:<web_tier_tag>,Values=<web_tier_tag_value> --query
"SecurityGroups[*].{GroupName:GroupName, IpPermissions:IpPermissions,
WebTierSecurityGroup:GroupId}" --output table
```

Remediation:

Using the Amazon unified command line interface:

- Remove the ingress rules for CIDR 0.0.0.0/0 (use the "WebTierSecurityGroup" element from Audit procedure):

```
aws ec2 revoke-security-group-ingress --group-id <web_tier_security_group> --
protocol tcp/udp --port <specific_port> --cidr 0.0.0.0/0
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-security-groups.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/revoke-security-group-ingress.html

6.21 Create the App tier ELB Security Group and ensure only accepts HTTP/HTTPS (Scored)

Profile Applicability:

- Level 1

Description:

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in the AWS Virtual Private Cloud (VPC), you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

Rationale:

The SG associated with the App tier ELB should allow connectivity from the security group associated with Web tier instances only for the HTTP (TCP 80) and HTTPS (TCP 443) ports.

The defaults for HTTP and HTTPS are used as an example, any other ports would apply depending on the application design.

Audit:

Using the Amazon unified command line interface:

- List the security group associated with the App tier ELB

```
aws elb describe-load-balancers --load-balancer-name <app_tier_elb> --query
"LoadBalancerDescriptions[*].{DNSName:DNSName,
AppTierELBSecurityGroups:SecurityGroups,
SourceSecurityGroup:SourceSecurityGroup}" --output table
```

- note the "AppTierELBSecurityGroups" from the output
- List the ingress rules for the above security group, and make sure that only HTTP (TCP 80) and HTTPS (TCP 443) traffic is allowed from Web tier Security Group:

```
aws ec2 describe-security-groups --group-ids  <app_tier_elb_security_group> --
--query "SecurityGroups[*].{GroupName:GroupName, IpPermissions:IpPermissions}"
--output table
```

Remediation:

Using the Amazon unified command line interface:

- First remove all the ingress rules for the security group associated with the App tier ELB:

```
aws ec2 describe-security-groups --group-id <app_tier_elb_security_group> --
query "SecurityGroups[0].IpPermissions" > /tmp/IpPermissions.json
aws ec2 revoke-security-group-ingress --group-id
<app_tier_elb_security_group> --ip-permissions file:///tmp/IpPermissions.json
```

- create locally the below json file containing ingress rules for HTTP (TCP 80) and HTTPS (TCP 443) ports only from <web_tier_security_group> and name it IpPermissions.json:

```
[
    {
        "PrefixListIds": [],
        "FromPort": 80,
        "IpRanges": [],
        "ToPort": 80,
        "IpProtocol": "tcp",
        "UserIdGroupPairs": [
            {
                "UserId": "<aws_account_number>",
                "GroupId": "<web_tier_security_group>"
            }
        ]
    },
    {
        "PrefixListIds": [],
        "FromPort": 443,
        "IpRanges": [],
        "ToPort": 443,
        "IpProtocol": "tcp",
        "UserIdGroupPairs": [
            {
                "UserId": "<aws_account_number>",
                "GroupId": "<web_tier_security_group>"
            }
        ]
    }
]
```

- Add to the security group associated with the App tier ELB the above ingress rules:

```
aws ec2 authorize-security-group-ingress --group-id
<app_tier_elb_security_group>  --ip-permissions
file:///PathTo/IpPermissions.json
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-security-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/revoke-security-group-ingress.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/authorize-security-group-ingress.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html

6.22 Create the App tier Security Group and ensure it allows inbound connections from App tier ELB Security Group for explicit ports (Scored)

Profile Applicability:

- Level 1

Description:

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in the AWS Virtual Private Cloud (VPC), you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

This is required for both the configured port and protocol for the listener on the back-end instance and the port and protocol used for the health check.

Rationale:

This protects the App-server tier from unauthorized access, it is recommended to add inbound security group rules that allow traffic for the specific application protocol and ports by referencing as source the security group associated with the App tier ELB.

Audit:

Using the Amazon unified command line interface:

- List the ingress rules for the above security group, and make sure that allows connections only from App tier ELB security group on specific ports:

```
aws ec2 describe-security-groups --filters
Name=tag:<app_tier_tag>,Values=<app_tier_tag_value> --query
"SecurityGroups[*].{GroupName:GroupName, IpPermissions:IpPermissions,
AppTierSecurityGroup:GroupId}" --output table
```

Remediation:

Using the Amazon unified command line interface:

- First remove all the ingress rules for the App tier security group (use the "AppTierSecurityGroup" element from Audit procedure):

```
aws ec2 describe-security-groups --group-id <app_tier_security_group> --query
"SecurityGroups[0].IpPermissions" > /tmp/IpPermissions.json
aws ec2 revoke-security-group-ingress --group-id <app_tier_security_group> --
ip-permissions file:///tmp/IpPermissions.json
```

- Add an ingress rule for a specific port, using --source-group option to specify the App tier ELB security group as the source of the connections:

```
aws ec2 authorize-security-group-ingress --group-id <app_tier_security_group>
--protocol tcp --port <specific_port> --source-group
<app_tier_elb_security_group>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/authorize-security-group-ingress.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/revoke-security-group-ingress.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-security-groups.html

6.23 Ensure App tier Security Group has no inbound rules for CIDR of 0 (Global Allow) (Scored)

Profile Applicability:

- Level 1

Description:

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in the AWS Virtual Private Cloud (VPC), you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

Rationale:

Considering any of the non-public tiers receive requests only either from the upper tier or from resources inside the same VPC, any inbound rules that allow traffic from any source (0.0.0.0/0) are not necessary and should be removed.

Audit:

Using the Amazon unified command line interface:

- List the ingress rules for the App tier security group, and make sure it has no inbound rules for CIDR of 0.0.0.0/0:

```
aws ec2 describe-security-groups --filters
Name=tag:<app_tier_tag>,Values=<app_tier_tag_value> --query
"SecurityGroups[*].{GroupName:GroupName, IpPermissions:IpPermissions,
AppTierSecurityGroup:GroupId}" --output table
```

Remediation:

Using the Amazon unified command line interface:

- Remove the ingress rules for CIDR 0.0.0.0/0 (use the "AppTierSecurityGroup" element from Audit procedure) :

```
aws ec2 revoke-security-group-ingress --group-id <app_tier_security_group> --
protocol tcp/udp --port <specific_port> --cidr 0.0.0.0/0
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-security-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html

6.24 Create the Data tier Security Group and ensure it allows inbound connections from App tier Security Group for explicit ports (Scored)

Profile Applicability:

- Level 1

Description:

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in the AWS Virtual Private Cloud (VPC), you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

The port for these inbound rules would depend on the Database engine used and the configured port.

The default values are:

MySQL - TCP 3306

MSSQL - TCP 1433

Oracle SQL - TCP 1521

PostgreSQL - TCP 5432

MariaDB - TCP 3306

Amazon Aurora DB - TCP 3306

Rationale:

This protects the Data tier from unauthorized access, it is recommended to add inbound security group rules that allow traffic for the specific database protocol and ports by referencing as source the security group associated with the App tier instances.

Audit:

Using the Amazon unified command line interface:

- Retrieve the Data tier security group configured for your RDS DB instance:

```
aws rds describe-db-instances --db-instance-identifier <your_db_instance> --
query "DBInstances[*].VpcSecurityGroups"
```

- List the ingress rules for the above security group, and make sure that allows connections only from App tier security group on specific ports:

```
aws ec2 describe-security-groups --group-ids  <data_tier_security_group> --
query "SecurityGroups[*].{GroupName:GroupName, IpPermissions:IpPermissions}"
--output table
```

Remediation:

Using the Amazon unified command line interface:

- First remove all the ingress rules for the security group configured for your RDS DB instance:

```
aws ec2 describe-security-groups --group-id <data_tier_security_group> --
query "SecurityGroups[0].IpPermissions" > /tmp/IpPermissions.json
aws ec2 revoke-security-group-ingress --group-id <data_tier_security_group> -
-ip-permissions file:///tmp/IpPermissions.json
```

- Add an ingress rule for a specific port, using --source-group option to specify the App tier security group as the source of the connections:

```
aws ec2 authorize-security-group-ingress --group-id
<data_tier_security_group> --protocol tcp --port <specific_port> --source-
group <app_tier_security_group>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-instances.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/authorize-security-group-ingress.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/revoke-security-group-ingress.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-security-groups.html

6.25 Ensure Data tier Security Group has no inbound rules for CIDR of 0 (Global Allow) (Scored)

Profile Applicability:

- Level 1

Description:

A *security group* acts as a virtual firewall for your instance to control inbound and outbound traffic. When you launch an instance in the AWS Virtual Private Cloud (VPC), you can assign the instance to up to five security groups. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could be assigned to a different set of security groups. If you don't specify a particular group at launch time, the instance is automatically assigned to the default security group for the VPC.

For each security group, you add *rules* that control the inbound traffic to instances, and a separate set of rules that control the outbound traffic.

Rationale:

Considering any of the non-public tiers receive requests only either from the upper tier or from resources inside the same VPC, any inbound rules that allow traffic from any source (0.0.0.0/0) are not necessary and should be removed.

Audit:

Using the Amazon unified command line interface:

- Retrieve the Data tier security group configured for your RDS DB instance:

```
aws rds describe-db-instances --db-instance-identifier <your_db_instance>
```

- List the ingress rules for the above security group, and make sure it has no inbound rules for CIDR of 0.0.0.0/0:

```
aws ec2 describe-security-groups --group-ids  <data_tier_security_group> --
query "SecurityGroups[*].{GroupName:GroupName, IpPermissions:IpPermissions}"
--output table
```

Remediation:

Using the Amazon unified command line interface:

- Remove the ingress rules for CIDR 0.0.0.0/0:

```
aws ec2 revoke-security-group-ingress --group-id <data_tier_security_group> -
-protocol tcp/udp --port <specific_port> --cidr 0.0.0.0/0
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-security-groups.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/revoke-security-group-ingress.html
3. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-instances.html

6.26 Ensure the App tier ELB is created as Internal (Scored)

Profile Applicability:

- Level 1

Description:

An internal load balancer routes traffic to your EC2 instances in private subnets using private IP addresses.

Create an internal load balancer and register the database servers with it. The web servers receive requests from the Internet-facing load balancer and send requests for the database servers to the internal load balancer. The database servers receive requests from the internal load balancer.

When an internal load balancer is created, it receives a public DNS name with the following form:

```
internal-name-123456789.region.elb.amazonaws.com
```

The DNS servers resolve the DNS name of your load balancer to the private IP addresses of the load balancer nodes for your internal load balancer. Each load balancer node is connected to the private IP addresses of the back-end instances that are in its Availability Zone using elastic network interfaces.

Rationale:

Creating the App tier ELB as internal will prevent access to the app tier from the Internet and will allow access from the Web tier instances.

Audit:

Using the Amazon unified command line interface:

- Check the Scheme of your App tier ELB, and make sure it is "internal":

```
aws elb describe-load-balancers --load-balancer-name <app_tier_elb> --query
"LoadBalancerDescriptions[*].{LoadBalancerName:LoadBalancerName,
Scheme:Scheme}"
```

Remediation:

Using the Amazon unified command line interface:

- Create new internal ELB for your App tier:

```
aws elb create-load-balancer --load-balancer-name <app_tier_elb> --scheme
internal --listeners <listener_config> --subnets <app_tier_subnet1>
<app_tier_subnet2> --security-groups <app_tier_elb_security_group>
```

- Register App tier instances with the new App tier ELB:

```
aws elb register-instances-with-load-balancer --load-balancer-name
<app_tier_elb> --instances <app_tier_instance1> <app_tier_instance2>
<app_tier_instance3>
```

References:

1. http://docs.aws.amazon.com/ElasticLoadBalancing/latest/DeveloperGuide/elb-internal-load-balancers.html
2. http://docs.aws.amazon.com/cli/latest/reference/elb/describe-load-balancers.html
3. http://docs.aws.amazon.com/cli/latest/reference/elb/create-load-balancer.html
4. http://docs.aws.amazon.com/cli/latest/reference/elb/register-instances-with-load-balancer.html

6.27 Ensure EC2 instances within Web Tier have no Elastic / Public IP addresses associated (Scored)

Profile Applicability:

- Level 1

Description:

All subnets have an attribute that determines whether instances launched into that subnet receive a public IP address. The public IP address is assigned to the default network interface (eth0). By default, instances launched into a default subnet are assigned a public IP address. A public IP address is mapped to the primary private IP address through network address translation (NAT).

An Elastic IP address is a static, public IP address designed for dynamic cloud computing. You can associate an Elastic IP address with any instance or network interface for your VPC. With an Elastic IP address, you can mask the failure of an instance by rapidly remapping the address to another instance in your VPC.

Rationale:

Without any Public or Elastic IP associated on the EC2 instance in the Web tier, no inbound traffic can reach the instance from the Internet.

Audit:

Using the Amazon unified command line interface:

- List the instance-id's, tags and public IP's of the EC2 instances from the application VPC that have a public IP associated, and check if there are any instances with Web tier tags listed:

```
aws ec2 describe-instances --filters "Name=vpc-id,Values=<application_vpc>" -
-query "Reservations[*].Instances[?PublicIpAddress !=
null].{InstanceId:InstanceId,
PublicIpAddresses:NetworkInterfaces[*].Association, SubnetId:SubnetId,
Tags:Tags}"
```

Remediation:

Using the Amazon unified command line interface:

- If in the above output the "IpOwnerId" is "amazon" the public Ip is not an Elastic IP and it cannot be manually disassociated from the instance after launch:

- Make sure that the Web tier subnet doesn't assign public Ip's at launch (run the command for all Web tier subnets)

```
aws ec2 modify-subnet-attribute --subnet-id <web_tier_subnet1> --no-map-public-ip-on-launch
```

- Create an AMI from the instance and launch a replacement instance in the same subnet

- If in the above output the "IpOwnerId" is an AWS account number, this is an Elastic IP and it can be disassociated:

```
aws ec2 disassociate-address --public-ip <elastic_ip_address>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/ec2/disassociate-address.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/modify-subnet-attribute.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html
4. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-ip-addressing.html

6.28 Ensure EC2 instances within App Tier have no Elastic / Public IP addresses associated (Scored)

Profile Applicability:

- Level 1

Description:

All subnets have an attribute that determines whether instances launched into that subnet receive a public IP address. The public IP address is assigned to the default network interface (eth0). By default, instances launched into a default subnet are assigned a public IP address. A public IP address is mapped to the primary private IP address through network address translation (NAT).

An Elastic IP address is a static, public IP address designed for dynamic cloud computing. You can associate an Elastic IP address with any instance or network interface for your VPC. With an Elastic IP address, you can mask the failure of an instance by rapidly remapping the address to another instance in your VPC.

Rationale:

Without any Public or Elastic IP associated on the EC2 instance in the App tier, no inbound traffic can reach the instance from the Internet.

Audit:

Using the Amazon unified command line interface:

- List the instance-id's, tags and public IP's of the EC2 instances from the application VPC that have a public IP associated, and check if there are any instances with App tier tags listed:

```
aws ec2 describe-instances --filters "Name=vpc-id,Values=<application_vpc>" --query "Reservations[*].Instances[?PublicIpAddress != null].{InstanceId:InstanceId, PublicIpAddresses:NetworkInterfaces[*].Association, SubnetId:SubnetId, Tags:Tags}"
```

Remediation:

Using the Amazon unified command line interface:

- If in the above output the "IpOwnerId" is "amazon" the public Ip is not an Elastic IP and it cannot be manually disassociated from the instance after launch:

- Make sure that the App tier subnet doesn't assign public Ip's at launch (run the command for all App tier subnets)

```
aws ec2 modify-subnet-attribute --subnet-id <app_tier_subnet1> --no-map-public-ip-on-launch
```

- Create an AMI from the instance and launch a replacement instance in the same subnet

- If in the above output the "IpOwnerId" is an AWS account number, this is an Elastic IP and it can be disassociated:

```
aws ec2 disassociate-address --public-ip <elastic_ip_address>
```

References:

1. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-ip-addressing.html
2. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-instances.html
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/modify-subnet-attribute.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/disassociate-address.html

6.29 Ensure EC2 instances within Data Tier have no Elastic / Public IP addresses associated (Scored)

Profile Applicability:

- Level 1

Description:

All subnets have an attribute that determines whether instances launched into that subnet receive a public IP address. The public IP address is assigned to the default network interface (eth0). By default, instances launched into a default subnet are assigned a public IP address. A public IP address is mapped to the primary private IP address through network address translation (NAT).

An Elastic IP address is a static, public IP address designed for dynamic cloud computing. You can associate an Elastic IP address with any instance or network interface for your VPC. With an Elastic IP address, you can mask the failure of an instance by rapidly remapping the address to another instance in your VPC.

Rationale:

Without any Public or Elastic IP associated on the EC2 instance in the Data tier, no inbound traffic can reach the instance from the Internet.

Audit:

Using the Amazon unified command line interface:

- List the instance-id's, tags and public IP's of the EC2 instances from the application VPC that have a public IP associated, and check if there are any instances with Data tier tags listed:

```
aws ec2 describe-instances --filters "Name=vpc-id,Values=<application_vpc>" -
-query "Reservations[*].Instances[?PublicIpAddress !=
null].{InstanceId:InstanceId,
PublicIpAddresses:NetworkInterfaces[*].Association, SubnetId:SubnetId,
Tags:Tags}"
```

Remediation:

Using the Amazon unified command line interface:

- If in the above output the "IpOwnerId" is "amazon" the public Ip is not an Elastic IP and it cannot be manually disassociated from the instance after launch:

- Make sure that the Data tier subnets doesn't assign public Ip's at launch (run the command for all Data tier subnets)

```
aws ec2 modify-subnet-attribute --subnet-id <data_tier_subnet1> --no-map-public-ip-on-launch
```

- Create an AMI from the instance and launch a replacement instance in the same subnet

- If in the above output the "IpOwnerId" is an AWS account number, this is an Elastic IP and it can be disassociated:

```
aws ec2 disassociate-address --public-ip <elastic_ip_address>
```

6.30 Ensure RDS Database is not publically accessible (Scored)

Profile Applicability:

- Level 2

Description:

Amazon Relational Database Service (RDS) is a managed relational database service which handles routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.

There are 6 database engines available for customer to run their database workloads on:

- Amazon Aurora (MySQL Compatible)
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- PostgreSQL

Customers can deploy RDS databases within a VPC through the configuration of:

- Subnet Group for RDS, this group will be used for deployment of single or Multi-AZ RDS instances.
- Network access through configuration of Security Groups for RDS
- Access from outside the VPC hosting the DB instance by enabling/disabling a Public IP address

Rationale:

Network access to the managed Data-Tier must be tightly controlled using Security Groups for RDS and non local accessibility of the DB instance.

Audit:

Using the Amazon unified command line interface:

- Check if your application DB instances are publicly available:

```
aws rds describe-db-instances --filters
Name=tag:<data_tier_tag>,Values=<data_tier_tag_value> --query
"DBInstances[*].{PubliclyAccessible:PubliclyAccessible,
DBInstanceIdentifier:DBInstanceIdentifier}"
```

Remediation:

Using the Amazon unified command line interface:

- Modify each publicly accessible DB instance, and make it private:

```
aws rds modify-db-instance --db-instance-identifier <your_db_instance> --no-
publicly-accessible
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-instance.html
2. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-instances.html

6.31 Don't use the default VPC (Scored)

Profile Applicability:

- Level 2

Description:

A default VPC is ready for you to use — you can immediately start launching instances into your default VPC without having to perform any additional configuration steps.

When we create a default VPC, AWS does the following to set it up:

1. Create a default subnet in each Availability Zone.
2. Create an Internet gateway and connect it to your default VPC.
3. Create a main route table for your default VPC with a rule that sends all traffic destined for the Internet to the Internet gateway.
4. Create a default security group and associate it with your default VPC.
5. Create a default network access control list (ACL) and associate it with your default VPC.
6. Associate the default DHCP options set for your AWS account with your default VPC.

Label this Default VPC "Do Not Use".

Rationale:

The default VPC comes with some default configuration that wouldn't meet the best practices, however if recommended setting are created or the default behavior is changed, this would still be considered ok.

Audit:

Using the Amazon Unified CLI:

- List the attributes of the VPC's in your account and note the value of the "IsDefault" attribute for the VPC where the application is deployed:

```
aws ec2 describe-vpcs --query "Vpcs[*].{VpcId:VpcId, IsDefault:IsDefault,
Tags:Tags}" --output table
```

Remediation:

Using the Amazon Unified CLI:

- Create a new VPC with the desired CIDR and migrate your application:

```
aws ec2 create-vpc --cidr-block <desired_cidr>
```

Impact:

The Default VPC can be deleted but only AWS Support can restore it. Don't delete it - just set a label to remind others not to use it.

References:

1. http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/default-vpc.html
2. https://aws.amazon.com/premiumsupport/knowledge-center/deleted-default-vpc/
3. http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-vpcs.html
4. http://docs.aws.amazon.com/cli/latest/reference/ec2/create-vpc.html

6.32 Ensure Auto-Scaling Launch Configuration for Web Tier is configured to use the Web Tier Security Group (Scored)

Profile Applicability:

- Level 1

Description:

Auto Scaling helps maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define.

You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances or can automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

These properties can be defined within the Auto-Scaling Group configuration.

Additional properties can be defined through the launch configuration such as:

- Instance Type
- Amazon Machine Image (Pre-configured Operating System Images - allows for O.S Hardening)
- IAM Role
- Security Groups

Rationale:

Instances within the Web Tier Auto-Scaling Group, are launched using the Security Group configured in the Auto-Scaling Launch Configuration. This Security Group allows only traffic specific to the Web Tier, and you must ensure that only this Web Tier Security Group is configured in the Launch Configuration.

Audit:

Using the Amazon Unified CLI:

- List the associated Launch Configuration of the Web Tier Auto-Scaling Group (note the value of "LaunchConfig" element):

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
<web_tier_autoscaling_group_name> --query
'AutoScalingGroups[*].{LaunchConfig:LaunchConfigurationName,ASG:AutoScalingGr
oupName}'
```

- Ensure actively used Launch Configuration found in the previous step is using the Web Tier Security Group (replace *<web_tier_launch_config>* with the Launch Configuration previously found):

```
aws autoscaling describe-launch-configurations --launch-configuration-names
<web_tier_launch_config> --query
'LaunchConfigurations[*].{LaunchConfig:LaunchConfigurationName,
SecurityGroups:SecurityGroups}'
```

Remediation:

Using the Amazon unified command line interface:

- Create new launch configuration for the Web tier using the Web Tier Security Group:

```
aws autoscaling create-launch-configuration --launch-configuration-name
<new_web_tier_launch_config> --image-id <web_tier_ami> --key-name
<your_key_pair> --security-groups <web_tier_security_group> --instance-type
<desired_instance_type> --iam-instance-profile <web_tier_instance_profile>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html

6.33 Ensure Auto-Scaling Launch Configuration for App Tier is configured to use the App Tier Security Group (Scored)

Profile Applicability:

- Level 1

Description:

Auto Scaling helps maintain application availability and allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define.

You can use Auto Scaling to help ensure that you are running your desired number of Amazon EC2 instances or can automatically increase the number of Amazon EC2 instances during demand spikes to maintain performance and decrease capacity during lulls to reduce costs.

These properties can be defined within the Auto-Scaling Group configuration.

Additional properties can be defined through the launch configuration such as:

- Instance Type
- Amazon Machine Image (Pre-configured Operating System Images - allows for O.S Hardening)
- IAM Role
- Security Groups

Rationale:

Instances within the App Tier Auto-Scaling Group, are launched using the Security Group configured in the Auto-Scaling Launch Configuration. This Security Group allows only traffic specific to the App Tier, and you must ensure that only this App Tier Security Group is configured in the Launch Configuration.

Audit:

Using the Amazon Unified CLI:

- List the associated Launch Configuration of the App Tier Auto-Scaling Group (note the value of "LaunchConfig" element):

```
aws autoscaling describe-auto-scaling-groups --auto-scaling-group-names
<app_tier_autoscaling_group_name> --query
'AutoScalingGroups[*].{LaunchConfig:LaunchConfigurationName,ASG:AutoScalingGr
oupName}'
```

- Ensure actively used Launch Configuration found in the previous step is using the App Tier Security Group (replace *<app_tier_launch_config>* with the Launch Configuration previously found):

```
aws autoscaling describe-launch-configurations --launch-configuration-names
<app_tier_launch_config> --query
'LaunchConfigurations[*].{LaunchConfig:LaunchConfigurationName,
SecurityGroups:SecurityGroups}'
```

Remediation:

Using the Amazon unified command line interface:

- Create new launch configuration for the App tier using the App Tier Security Group :

```
aws autoscaling create-launch-configuration --launch-configuration-name
<new_app_tier_launch_config> --image-id <app_tier_ami> --key-name
<your_key_pair> --security-groups <app_tier_security_group> --instance-type
<desired_instance_type> --iam-instance-profile <app_tier_instance_profile>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-launch-configurations.html
2. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/create-launch-configuration.html
3. http://docs.aws.amazon.com/cli/latest/reference/autoscaling/describe-auto-scaling-groups.html

6.34 Ensure RDS Database is configured to use the Data Tier Security Group (Scored)

Profile Applicability:

- Level 1

Description:

Amazon Relational Database Service (RDS) is a managed relational database service which handles routine database tasks such as provisioning, patching, backup, recovery, failure detection, and repair.

There are 6 database engines available for customer to run their database workloads on:

- Amazon Aurora (MySQL Compatible)
- MySQL
- MariaDB
- Oracle
- Microsoft SQL Server
- PostgreSQL

Customers can deploy RDS databases within a VPC through the configuration of:

- Subnet Group for RDS, this group will be used for deployment of single or Multi-AZ RDS instances.
- Network access through configuration of Security Groups for RDS
- Access from outside the VPC hosting the DB instance by enabling/disabling a Public IP address

Rationale:

Network access to the managed Data-Tier must be tightly controlled using Security Groups for RDS and non local accessibility of the DB instance.

Audit:

Using the Amazon unified command line interface:

- Check if your application DB instances are configured to use the Data Tier Security Group:

```
aws rds describe-db-instances --filters
Name=tag:<data_tier_tag>,Values=<data_tier_tag_value> --query
```

```
"DBInstances[*].{VpcSecurityGroups:VpcSecurityGroups,
DBInstanceIdentifier:DBInstanceIdentifier}"
```

Remediation:

Using the Amazon unified command line interface:

- Modify each non-compliant DB instance, and configure it to use the Data Tier Security Group:

```
aws rds modify-db-instance --db-instance-identifier <your_db_instance> --vpc-
security-group-ids <data_tier_security_group>
```

References:

1. http://docs.aws.amazon.com/cli/latest/reference/rds/describe-db-instances.html
2. http://docs.aws.amazon.com/cli/latest/reference/rds/modify-db-instance.html

7 Appendix - Variables

A number of variables are represented in this benchmark which need to be replaced with proper values for your environment. The following is a list of all variables which will need considering:

```
<app_tier_elb_security_group> 3.15, 6.21, 6.22, 6.26
<admin_policy_arn> 2.7, 2.8
<admin_policy_name> 2.8
<admin_policy_version> 2.8
<app_tier_autoscaling_group_name> 3.4, 3.15, 6.33
<app_tier_elb> 1.12, 1.13, 1.14, 3.10, 3.15, 6.21, 6.26
<app_tier_iam_role> 2.2, 2.4
<app_tier_instance_profile> 2.2, 2.4, 2.5, 3.4, 5.9, 6.18, 6.33
<app_tier_instance1> 6.26
<app_tier_kms_alias> 1.2
<app_tier_kms_key> 1.2, 1.6, 5.11
<app_tier_log_group> 5.5, 5.7
<app_tier_security_group> 2.6, 3.4, 5.9, 6.18, 6.22, 6.23, 6.24, 6.32
<app_tier_subnet1> 3.15, 6.7, 6.14, 6.15, 6.26, 6.28
<app_tier_subnet2> 3.15, 6.7, 6.15, 6.26
<app_tier_tag_value> 1.6, 1.8, 2.4, 5.11, 6.7, 6.15, 6.22, 6.23
<app_tier_tag> 1.6, 1.8, 2.4, 5.11, 6.7, 6.15, 6.22, 6.23
<application_az> 1.5, 1.6, 3.2
<application_cfn_distribution_etag> 1.17, 3.12, 3.13, 5.3, 6.4
<application_cfn_distribution_id> 1.17, 3.12, 3.13, 5.3, 6.4
<application_region> 1.5, 1.6, 4.6
<application_vpc> 3.2, 6.5, 6.6, 6.7, 6.8, 6.10, 6.11, 6.12, 6.27, 6.28, 6.29
<autoscaling_group_name> 3.1, 3.2
<aws_account_number> 6.21
<backup_retention_period> 3.8
<billing_alarm_name> 4.8
<blacklist|whitelist> 6.4
```

<cloudwatch_agent_config_file> 5.8, 5.9
<country_code_1> 6.4
<country_code_2> 6.4
<data_tier_kms_alias> 1.3
<data_tier_kms_key> 1.3, 1.4
<data_tier_security_group> 6.24, 6.25, 6.34
<data_tier_subnet1> 6.16, 6.29
<data_tier_subnet2> 6.16
<data_tier_tag_value> 3.5, 3.6, 3.8, 6.3, 6.16, 6.30, 6.34
<data_tier_tag> 3.5, 3.6, 3.8, 6.3, 6.16, 6.30, 6.34
<db_snapshot> 1.4
<desired_cidr> 6.5, 6.6, 6.7, 6.8, 6.31
<desired_evaluation_periods> 4.7
<desired_instance_type> 2.5, 2.6, 3.3, 3.4, 5.8, 5.9, 6.18, 6.31, 6.33
<desired_operator> 4.7
<desired_period> 4.7
<desired_statistic> 4.7
<desired_threshold> 4.7
<elastic_ip_address> 6.27, 6.28, 6.29
<elastic_ip_allocation> 6.10
<elb_name> 3.1, 5.2
<encrypted_db_snapshot> 1.4
<encrypted_ebs_snapshot> 1.5, 1.6
<encrypted_ebs_volume> 1.5, 1.6
<estimated_charges> 4.8
<events_source_ids> 4.3, 4.5
<iam_admin_group_name> 2.7
<iam_policy_arn> 2.1, 2.2
<iam_policy_name> 2.1, 2.2
<iam_policy_version> 2.1, 2.2
<iam_user> 2.9, 2.10
<latest_ssl_policy> 1.10, 1.13
<listener_config> 3.1, 3.14, 3.15, 6.26
<log_retention_period> 5.6, 5.7
<private_subnet1> 6.12
<private_subnet2> 6.12
<protocol_for_sns> 4.1, 4.2
<public_image_id> 1.7, 1.8
<public_subnet1> (for NAT; can be the same as the one for elb) 6.10, 6.11
<public_subnet2> (for NAT; can be the same as the one for elb) 6.10, 6.11
<public_tier_tag_value> 6.5
<public_tier_tag> 6.5
<rds_event_subscription> 4.3, 4.5
<rds_events> 4.3, 4.5
<route_table_id> 6.11, 6.12, 6.13, 6.14, 6.15, 6.16
<s3_bucket_name> 1.16, 3.11, 5.3, 5.8, 5.9
<sns_subscription_endpoints> 4.1, 4.2
<sns_topic_arn> 2.9, 2.10, 4.1, 4.2, 4.3, 4.5, 4.7, 4.8
<sns_topic_name> 4.1, 4.2
<specific_port> 6.19, 6.20, 6.22, 6.23, 6.24, 6.25
<ssl_certificate_arn> 1.9, 1.11, 1.12, 1.14, 1.15
<ssl_certificate_name> 1.15
<unencrypted_ebs_snapshot> 1.5, 1.6
<unencrypted_ebs_volume> 1.5, 1.6
<vpc_flow_log_alarm_name> 4.7
<vpc_flow_log_filter_name> 4.6
<vpc_flow_log_group_name> 4.6

```
<vpc_flow_log_metric_name> 4.6, 4.7
<vpc_igw> 6.11, 6.13
<vpc_nat_gw> 6.12, 6.14, 6.15
<web_tier_ami> 2.5, 3.3, 5.8, 6.18, 6.32
<web_tier_autoscaling_group_name> 3.3, 3.14, 6.32
<web_tier_elb_security_group> 3.14, 6.19
<web_tier_elb_subnet1> 3.14, 6.5, 6.13
<web_tier_elb_subnet2> 3.14, 6.5, 6.13
<web_tier_elb> 1.9, 1.10, 1.11, 3.9, 3.14, 6.5, 6.13, 6.18
<web_tier_iam_role> 2.1, 2.3
<web_tier_instance_profile> 2.1, 2.3, 2.5, 3.3,  5.8, 6.18, 6.32
<web_tier_kms_alias> 1.1
<web_tier_kms_key> 1.1, 1.5, 5.10
<web_tier_launch_config> 2.5, 3.3, 5.8, 6.18, 6.32
<web_tier_log_group> 5.4, 5.6
<web_tier_security_group> 2.5, 5.8, 6.18, 6.19, 6.21, 6.31
<web_tier_subnet1> 6.6, 6.14, 6.27
<web_tier_subnet2> 6.6, 6.14
<web_tier_tag_value> 1.5, 1.7, 2.3, 5.10, 6.6, 6.14, 6.19, 6.20
<web_tier_tag> 1.5, 1.7, 2.3, 5.10, 6.6, 6.14, 6.19, 6.20
<your_db_instance> 1.4, 3.5, 3.6, 3.8, 6.24, 6.25, 6.30, 6.34
<your_key_pair> 2.5, 2.6, 3.3, 3.4, 5.8, 5.9, 6.18, 6.31, 6.33
```

Appendix: Summary Table

Control		Set Correctly	
		Yes	No
1	**Data Protection**		
1.1	Ensure a customer created Customer Master Key (CMK) is created for the Web-tier (Scored)	☐	☐
1.2	Ensure a customer created Customer Master Key (CMK) is created for the App-tier (Scored)	☐	☐
1.3	Ensure a customer created Customer Master Key (CMK) is created for the Database-Tier (Scored)	☐	☐
1.4	Ensure Databases running on RDS have encryption at rest enabled (Scored)	☐	☐
1.5	Ensure all EBS volumes for Web-Tier are encrypted (Scored)	☐	☐
1.6	Ensure all EBS volumes for App-Tier are encrypted (Scored)	☐	☐
1.7	Ensure all Customer owned Amazon Machine Images for Web Tier are not shared publicly (Scored)	☐	☐
1.8	Ensure all Customer owned Amazon Machine Images for Application Tier are not shared publicly (Scored)	☐	☐
1.9	Ensure Web Tier ELB have SSL/TLS Certificate attached (Scored)	☐	☐
1.10	Ensure Web Tier ELB have the latest SSL Security Policies configured (Scored)	☐	☐
1.11	Ensure Web Tier ELB is using HTTPS listener (Scored)	☐	☐
1.12	Ensure App Tier ELB have SSL\TLS Certificate attached (Scored)	☐	☐
1.13	Ensure App Tier ELB have the latest SSL Security Policies configured (Scored)	☐	☐
1.14	Ensure App Tier ELB is using HTTPS listener (Scored)	☐	☐
1.15	Ensure all Public Web Tier SSL\TLS certificates are >30 days from Expiration (Scored)	☐	☐
1.16	Ensure all S3 buckets have policy to require server-side and in transit encryption for all objects stored in bucket. (Scored)	☐	☐
1.17	Ensure CloudFront to Origin connection is configured using TLS1.1+ as the SSL\TLS protocol (Scored)	☐	☐
2	**Identity and Access Management**		
2.1	Ensure IAM Policy for EC2 IAM Roles for Web tier is configured (Scored)	☐	☐
2.2	Ensure IAM Policy for EC2 IAM Roles for App tier is configured (Scored)	☐	☐
2.3	Ensure an IAM Role for Amazon EC2 is created for Web Tier (Scored)	☐	☐

2.4	Ensure an IAM Role for Amazon EC2 is created for App Tier (Scored)	☐	☐
2.5	Ensure AutoScaling Group Launch Configuration for Web Tier is configured to use a customer created Web-Tier IAM Role (Scored)	☐	☐
2.6	Ensure AutoScaling Group Launch Configuration for App Tier is configured to use an App-Tier IAM Role (Scored)	☐	☐
2.7	Ensure an IAM group for administration purposes is created (Scored)	☐	☐
2.8	Ensure an IAM policy that allows admin privileges for all services used is created (Scored)	☐	☐
2.9	Ensure SNS Topics do not Allow 'Everyone' To Publish (Scored)	☐	☐
2.10	Ensure SNS Topics do not Allow 'Everyone' To Subscribe (Scored)	☐	☐
3	**Business Continuity**		
3.1	Ensure each Auto-Scaling Group has an associated Elastic Load Balancer (Scored)	☐	☐
3.2	Ensure each Auto-Scaling Group is configured for multiple Availability Zones (Scored)	☐	☐
3.3	Ensure Auto-Scaling Launch Configuration for Web-Tier is configured to use an approved Amazon Machine Image (Scored)	☐	☐
3.4	Ensure Auto-Scaling Launch Configuration for App-Tier is configured to use an approved Amazon Machine Image (Scored)	☐	☐
3.5	Ensure Relational Database Service is Multi-AZ Enabled (Scored)	☐	☐
3.6	Ensure Relational Database Service Instances have Auto Minor Version Upgrade Enabled (Scored)	☐	☐
3.8	Ensure Relational Database Service backup retention policy is set (Scored)	☐	☐
3.9	Ensure Web Tier Elastic Load Balancer has application layer Health Check Configured (Scored)	☐	☐
3.10	Ensure App Tier Elastic Load Balancer has application layer Health Check Configured (Scored)	☐	☐
3.11	Ensure S3 buckets have versioning enabled (Scored)	☐	☐
3.12	Configure HTTP to HTTPS Redirects with a CloudFront Viewer Protocol Policy (Scored)	☐	☐
3.13	Ensure all CloudFront Distributions require HTTPS between CloudFront and your Web-Tier ELB origin (Scored)	☐	☐
3.14	Ensure Web Tier Auto-Scaling Group has an associated Elastic Load Balancer (Scored)	☐	☐
3.15	Ensure App Tier Auto-Scaling Group has an associated Elastic	☐	☐

	Load Balancer (Scored)		
4	**Event Monitoring and Response**		
4.1	Ensure a SNS topic is created for sending out notifications from Cloudtwatch Alarms and Auto-Scaling Groups (Scored)	☐	☐
4.2	Ensure a SNS topic is created for sending out notifications from RDS events (Scored)	☐	☐
4.3	Ensure RDS event subscriptions are enabled for Instance level events (Scored)	☐	☐
4.4	Ensure RDS event subscriptions are enabled for DB security groups (Scored)	☐	☐
4.6	Ensure that a log metric filter for the Cloudwatch group assigned to the "VPC Flow Logs" is created (Scored)	☐	☐
4.7	Ensure that a Cloudtwach Alarm is created for the "VPC Flow Logs" metric filter, and an Alarm Action is configured (Scored)	☐	☐
4.8	Ensure Billing Alerts are enabled for increments of X spend (Scored)	☐	☐
5	**Audit and Logging**		
5.1	Ensure all resources are correctly tagged (Not Scored)	☐	☐
5.2	Ensure AWS Elastic Load Balancer logging is enabled (Scored)	☐	☐
5.3	Ensure AWS Cloudfront Logging is enabled (Scored)	☐	☐
5.4	Ensure Cloudwatch Log Group is created for Web Tier (Scored)	☐	☐
5.5	Ensure Cloudwatch Log Group is created for App Tier (Scored)	☐	☐
5.6	Ensure Cloudwatch Log Group for Web Tier has a retention period (Scored)	☐	☐
5.7	Ensure Cloudwatch Log Group for App Tier has a retention period (Scored)	☐	☐
5.8	Ensure an agent for AWS Cloudwatch Logs is installed within Auto-Scaling Group for Web-Tier (Not Scored)	☐	☐
5.9	Ensure an agent for AWS Cloudwatch Logs is installed within Auto-Scaling Group for App-Tier (Not Scored)	☐	☐
5.10	Ensure an AWS Managed Config Rule for encrypted volumes is applied to Web Tier (Scored)	☐	☐
5.11	Ensure an AWS Managed Config Rule for encrypted volumes is applied to App Tier (Scored)	☐	☐
5.12	Ensure an AWS Managed Config Rule for EIPs attached to EC2 instances within VPC (Scored)	☐	☐
6	**Networking**		
6.1	Ensure Root Domain Alias Record Points to ELB (Scored)	☐	☐
6.2	Ensure a DNS alias record for the root domain (Scored)	☐	☐
6.3	Use CloudFront Content Distribution Network (Scored)	☐	☐
6.4	Ensure Geo-Restriction is enabled within Cloudfront	☐	☐

	Distribution (Scored)		
6.5	Ensure subnets for the Web tier ELB are created (Scored)	☐	☐
6.6	Ensure subnets for the Web tier are created (Scored)	☐	☐
6.7	Ensure subnets for the App tier are created (Scored)	☐	☐
6.8	Ensure subnets for the Data tier are created (Scored)	☐	☐
6.9	Ensure Elastic IPs for the NAT Gateways are allocated (Scored)	☐	☐
6.10	Ensure NAT Gateways are created in at least 2 Availability Zones (Scored)	☐	☐
6.11	Ensure a route table for the public subnets is created (Scored)	☐	☐
6.12	Ensure a route table for the private subnets is created (Scored)	☐	☐
6.13	Ensure Routing Table associated with Web tier ELB subnet have the default route (0.0.0.0/0) defined to allow connectivity to the VPC Internet Gateway (IGW) (Scored)	☐	☐
6.14	Ensure Routing Table associated with Web tier subnet have the default route (0.0.0.0/0) defined to allow connectivity to the VPC NAT Gateway (Scored)	☐	☐
6.15	Ensure Routing Table associated with App tier subnet have the default route (0.0.0.0/0) defined to allow connectivity to the VPC NAT Gateway (Scored)	☐	☐
6.16	Ensure Routing Table associated with Data tier subnet have NO default route (0.0.0.0/0) defined to allow connectivity to the VPC NAT Gateway (Scored)	☐	☐
6.17	Use a Web-Tier ELB Security Group to accept only HTTP/HTTPS (Scored)	☐	☐
6.18	Ensure Web tier ELB Security Group is not used in the Auto Scaling launch configuration of any other tier (Web, App) (Scored)	☐	☐
6.19	Create the Web tier Security Group and ensure it allows inbound connections from Web tier ELB Security Group for explicit ports (Scored)	☐	☐
6.20	Ensure Web tier Security Group has no inbound rules for CIDR of 0 (Global Allow) (Scored)	☐	☐
6.21	Create the App tier ELB Security Group and ensure only accepts HTTP/HTTPS (Scored)	☐	☐
6.22	Create the App tier Security Group and ensure it allows inbound connections from App tier ELB Security Group for explicit ports (Scored)	☐	☐
6.23	Ensure App tier Security Group has no inbound rules for CIDR of 0 (Global Allow) (Scored)	☐	☐
6.24	Create the Data tier Security Group and ensure it allows inbound connections from App tier Security Group for explicit ports (Scored)	☐	☐

6.25	Ensure Data tier Security Group has no inbound rules for CIDR of 0 (Global Allow) (Scored)	☐	☐
6.26	Ensure the App tier ELB is created as Internal (Scored)	☐	☐
6.27	Ensure EC2 instances within Web Tier have no Elastic / Public IP addresses associated (Scored)	☐	☐
6.28	Ensure EC2 instances within App Tier have no Elastic / Public IP addresses associated (Scored)	☐	☐
6.29	Ensure EC2 instances within Data Tier have no Elastic / Public IP addresses associated (Scored)	☐	☐
6.30	Ensure RDS Database is not publically accessible (Scored)	☐	☐
6.31	Don't use the default VPC (Scored)	☐	☐
6.32	Ensure Auto-Scaling Launch Configuration for Web Tier is configured to use the Web Tier Security Group (Scored)	☐	☐
6.33	Ensure Auto-Scaling Launch Configuration for App Tier is configured to use the App Tier Security Group (Scored)	☐	☐
6.34	Ensure RDS Database is configured to use the Data Tier Security Group (Scored)	☐	☐
7	**Appendix - Variables**		

Appendix: Change History

Date	Version	Changes for this version
11-09-2016	1.0.0	Initial Release