Technical Countermeasure Report

User Accounts

Report generated by:	Administrator admin
Unique ID:	user-accounts
Workflow State:	

Index

Summary

Architectural Diagrams

Required Countermeasures
- Component: API GW
- Component: MySQL

Implemented Countermeasures
- Component: API GW
- Component: MySQL

Rejected Countermeasures
- Component: API GW

Countermeasure Test Results
- Failed
  - Component: API GW
  - Component: MySQL
- Passed
  - Component: API GW
- Not tested
  - Component: ELB - Elastic Load Balancer
  - Component: MySQL

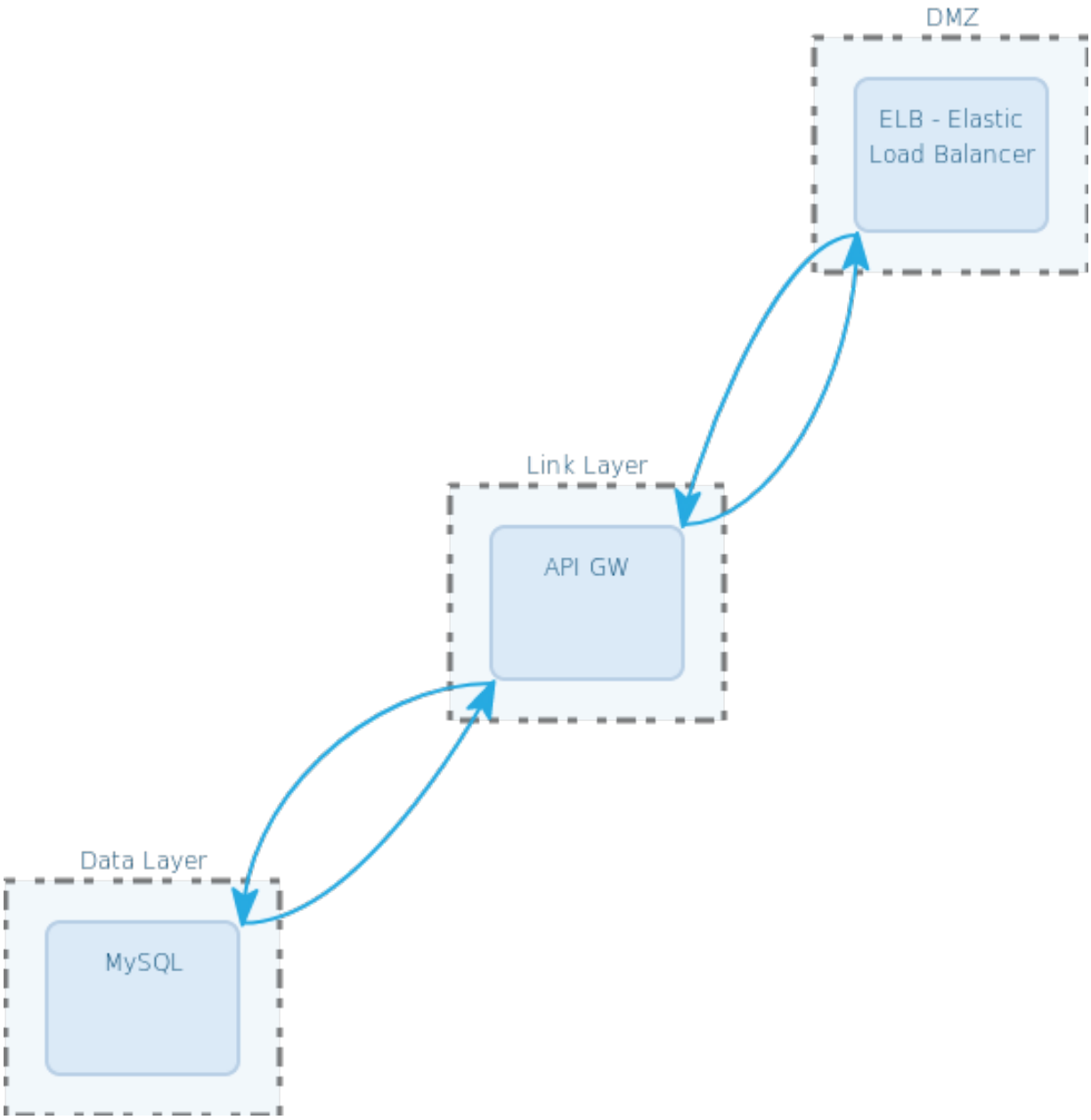
Appendix A: Countermeasure Details
- Component: API GW
- Component: ELB - Elastic Load Balancer
- Component: MySQL

Summary

Shown below is a brief description of the product and summary analysis of the risks.

Product name:	User Accounts
Unique ID:	user-accounts
Product description:	
Business unit:	bu_user_admin
Owner:	Administrator admin

Architectural Diagrams



Filename: irius-risk-diagram-architecture-image.png
Username: admin **Date uploaded:** 09-May-2019 15:20:22

Required Countermeasures

Component: API GW

Countermeasure name	Test result	Related threats
Log details of user actions within the system	Not Tested	• Attacks against the authentication system may go undetected
Log and reject all data validation failures	Not Tested	• Attacks against the authentication system may go undetected
Encrypt data between the client and server/service	Not Tested	• Attackers gain control of the connection through a Man In The Middle attack
Use a synchronised time source	Not Tested	• Attacks against the authentication system may go undetected
Validate all data received from the client side	Not Tested	• Attackers gain access to the system through Server Side Code Injection
Log the backend TLS connection failures	Not Tested	• Attacks against the authentication system may go undetected
Ensure the integrity of the logging system	Not Tested	• An attacker injects, manipulates or forges malicious log entries in the log file, allowing him to mislead a log audit, cover traces of attack, or perform other malicious actions
Limit the number of accounts with privileges allowing modification and/or deletion of audit logs files	Failed	• An attacker injects, manipulates or forges malicious log entries in the log file, allowing him to mislead a log audit, cover traces of attack, or perform other malicious actions
Ensure that the client-side and the server-side are using the same encoding style	Not Tested	• Attackers gain access to the system through Server Side Code Injection

Component: MySQL

Countermeasure name	Test result	Related threats
Use prepared statements for all database queries	Not Tested	• Attackers gain unauthorised access to data and/or systems through SQL Injection attacks
Apply required security patches to the service	Not Tested	• Attackers gain access to unauthorised data by exploiting vulnerabilities in the service

Implemented Countermeasures

Component: API GW

Countermeasure name	Test result	Related threats
Escape meta-characters from un-trusted data	Passed	• An attacker injects, manipulates or forges malicious log entries in the log file, allowing him to mislead a log audit, cover traces of attack, or perform other malicious actions
Do not write secrets to the log files	Not Tested	• Data leakage or disclosure to unauthorized parties
Develop a log retention policy	Not Tested	• Data leakage or disclosure to unauthorized parties
Restrict actions of users that follow unusual patterns.	Not Tested	• Attackers subvert the intended workflow of the application in order to perform unauthorised operations

Component: MySQL

Countermeasure name	Test result	Related threats
Require authentication before presenting restricted data	Failed	• Attackers obtain unauthorised access by connecting directly to the service
Access the data store from an account with the least privileges necessary	Not Tested	• Attackers who compromise the application or application server could directly access and modify the data store

Rejected Countermeasures

Component: API GW

Countermeasure name	Test result	Related threats	Reason
Implement application and network rate limiting	Not Tested	• Denial of service through resource exhaustion	This Component is not planned to be behind the WAF, we cannot implement this on a feasible way.
Detect and notify the usage of automated tools or unusual behavior	Not Tested	• Attackers subvert the intended workflow of the application in order to perform unauthorised operations	This Component is not planned to be behind the WAF, we cannot apply this countermeasure.

Countermeasure Test Results

Failed

The below table shows all countermeasures with failed test results.

Component: API GW

Name	Description
Limit the number of accounts with privileges allowing modification and/or deletion of audit logs files	Limit the number of account with privileges to modify and/or delete audit logs files.

Component: MySQL

Name	Description
Require authentication before presenting restricted data	The application should ensure users have undergone an Identification and Verification (ID&V) process before allowing access to secret, sensitive or otherwise restricted data. For less sensitive but still restricted data, simple verification of the location of the user may suffice (e.g. IP restrictions). • For non-sensitive but non-public data, access could be restricted by IP address, for example limiting access to internal networks, workstations, or gateways • For more sensitive data, TLS client-side certificates may be appropriate • Where secret or other sensitive data is handled, a full authentication process to identify and validate users with single or multi-factor authentication may be required

Passed

The below table shows all countermeasures with passed test results.

Component: API GW

Name	Description
Escape meta-characters from un-trusted data	If untrusted data, including any data received from the client side of a connection is directly written to a log file, then this data could contain newline or other meta-characters that would allow an attacker to forge log entries. Such meta-characters should first be escaped or removed before the data is written to the logging system.

Not tested

The below shows all countermeasures with not tested results.

Component: API GW

Name	Description
Log details of user actions within the system	To maintain proper accountability, logs should be maintained with sufficient information to track user actions within the system. These logs should be forensically sound, non-repudiable, and contain comprehensive details about activity. While the exact data for an event may vary, the following should be captured at a minimum: • Timestamps against a proven external source (e.g. an NTP server) • Origin, with this field we mark if the logs are provided by a trusted or untrusted source. • Event, status, and/or error codes (with sensitive data masked as appropriate or not introduced in logs) • Service, command, application or function name and details • User or system account associated with an event • Devices used (e.g. source and destination IPs, terminal session ID, web browser, etc) Source: https://security.berkeley.edu/security-audit-logging-guideline
Log and reject all data validation failures	Data validation failures, together with access control violations, are symptomatic of malicious activity where an attacker is attempting to subvert the protections in place. It is therefore likely that unexpected input detected by the application relates to an attack. Rejecting and logging such activity, and ideally terminating the session, increases the likelihood of detecting and inhibiting structured attacks against the application. • Log all validation failures when rejecting requests. • Ensure logged data is appropriately sanitized and encoded to prevent attacks against the logs and subsequent access to them. • Terminate the offending user session to inhibit further attack. • Ensure errors returned to the client-side are generic to prevent an attacker enumerating the defenses in place or gaining knowledge about the back-end.
Encrypt data between the client and server/service	Data passed between the client and server should be protected by encryption in transit. • Implement cryptographically strong TLS end-to-end encryption between the client and server, terminating within a secure environment on the server-side. • Consider use of client certificates to prevent interception of (or man-in-the-middle attacks on) the encrypted connection. • Alternatively, asymmetric (public-key) encryption could be utilized, although a recognized, proven, and tested implementation/library should be used
Do not write secrets to the log files	The logs may be accessed by attackers and in order to protect sensitive data, no such sensitive data should be included in the logs
Prevent unauthorised access to source code through the service	Access to the source-code for the application can aid an attacker in determined bugs or vulnerabilities in the code or logic. For closed-source projects it is therefore important to control and restrict access to the source. Application services may unexpectedly expose code, for example a service providing files to a user could be manipulated to access source code if implemented insecurely. • Ensure that source code is not inadvertently disclosed through misconfiguration or vulnerabilities in the service. • Check that configuration files are not downloadable directly from the service, and cannot be read and viewed through the service itself. • Ensure backups, operating system, and version control artifacts do not expose code.

Use a synchronised time source	In order to correlate logs and data from different internal and external systems, and to preserve forensic quality of the logs, it is important a unified and trusted synchronized time source is used throughout the environment. • Servers should be synchronize to an internal or external NTP server • The centralized source should in turn use (or be) a trusted central time source. This control is critical in identifying application events (including attacks) through logging, and in conduction post-event analysis, in particular to track the whole user (or attacker) journey through the system should it be compromised. It is good practice to use the concept of Indicators of Compromise (IoC) should be used to detect possible situations in which the system has been compromised and to give an appropriate response. IoCs are often tracked through logs, and accurate time is often essential.
Implement application and network rate limiting	A number of attacks rely on brute-force techniques to send large volumes of requests to enumerate or attempt to exploit flaws in an application, for example, sending common passwords to multiple target accounts within an application. By profiling normal traffic volumes, and applying rate limiting, the application can be built to actively mitigate such attacks. • Connection rate-limiting based on the source IP address can be used to restrict attacks against the authentication or registration systems. Multiple failures (or attempts) from a single IP should result in temporarily blocking or dropping traffic from the source. Note however that some corporate and ISP environment may place multiple valid and discrete clients behind the same IP address, resulting in false-positives. • Attackers may use botnets and other IP masking techniques to deliver attacks from multiple sources to avoid IP based rate-limiting. To mitigate this class of attack, Indicators of Compromise should be monitored (for example a higher rate of login failures than usual), and appropriate actions taken. For example, when the application detects active brute-force attacks, a Web Application Firewall (WAF) or other intermediate devices could be used to block attacks sharing a signature from pattern matching or deep packet inspection (e.g. HTTP headers or common passwords across multiple accounts). Similarly, the application could respond by requiring a CAPTCHA, cookie, or Javascript challenge when an attack is detected. **Remediation:** Implement the mechanisms to lockout accounts: • When the application detects a set number of failure login attempts, the account shall be locked for a certain time period. • When the application detects that an account is locked more times than usual, this account shall be disabled. A disabled account shall only be restored by an administrator. • When the application detects active brute-force attacks, the application shall require a CAPTCHA, cookie, or JavaScript challenge.

Validate all data received from the client side	All data received from the client-side should be considered tainted and a potential risk, regardless of the source or transport method. For example, while hidden form fields, cookies, or other headers may be obfuscated from a user, along with parameters passed in ViewStates or other encapsulated forms, these can be modified by the user at the client-side in memory, or in transit on the network. Similarly, data passed from binary or compiled components can be modified in situ or in transit. *Furthermore, encryption only secures the data in transit between the two ends of the encrypted tunnel (one end of which is typically controlled by the client); data passing through the link may still be malicious.* As such, all data from the client side must be subjected to strict validation, sanitization, and encoding against expected syntactic and semantic criteria. • Define a specification of the data that is expected at each input; both the syntax (e.g. alphanumeric only) and semantics (e.g. a word of between 1 and 25 characters, or a specific list). As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." • Implement a 'known good' or white-list approach, where only inputs that meet the strict criteria for each input are accepted, and reject, transform, or encapsulate any non-compliant data. • While useful for identifying malicious content, do not rely on looking for specific malformed or attack payloads (blacklists). It is almost impossible to maintain a comprehensive and accurate blacklist due to the complexity and evolving nature of attacks, opportunities to obfuscate payloads, and changes to the code's execution environment. As noted, blacklists can be useful for detecting and logging potential attacks, or determining which inputs are so malformed that they should be rejected outright. • Validate all data received from the client, including values such as HTTP headers and cookie values if these are used as input on the server side, X- headers, and other platform specific data objects passed between the client and server.
Develop a log retention policy	Develop a log retention policy to identify storage requirements for device logs and implement procedures to ensure that the audit logs are available for a security response in the case of incident or investigation. The audit logs must be collected for the last 30 days in easily accessible storage media. Older logs should be archived in a protected storage and should be accessible in the future as required for incidents or investigations.
Log the backend TLS connection failures	Implement functionality to record backend TLS connection failures and include these in the logs.
Ensure the integrity of the logging system	Ensure Log integrity for the application generated logs, such as storing logs on write-once media, forwarding a copy of the logs to a centralized SIEM or generating message digests for each log file.
Detect and notify the usage of automated tools or unusual behavior	Don't allow users to manipulate a system or guess its behavior based on input or output timing and detect the usage of automated tools or unusual behavior, such as actions not performed in reasonable "human time" or other abnormal time patterns. When the usage of automated tools is detected, the application shall respond with denying the access and notifying the security group.
Restrict actions of users that follow unusual patterns.	Restrict actions that users can do outside of the approved/required business process flow. This is important because without this safeguard in place attackers may be able to bypass or circumvent work-flows and checks allowing them to prematurely enter or skip required sections of the application potentially allowing action/transaction to be completed without successfully completing the entire business process, leaving the system with incomplete back-end tracking information.
Ensure that the client-side and the server-side are using the same encoding style	Ensure that the client-side and the server-side are using the same encoding style.

Component: ELB - Elastic Load Balancer

Name	Description
Use the Perfect Forward Secrecy feature	For greater communication privacy Elastic Load Balancing allows the use of Perfect Forward Secrecy. This feature provides additional safeguards against eavesdropping on encrypted data, through the use of a unique random session key, and therefore prevents the decoding of captured data, even if the secret long-term key is compromised. **To begin using Perfect Forward Secrecy:** Configure your load balancer with the newly added Elliptic Curve Cryptography (ECDHE) cipher suites.
Select the Server Order Preference option	Within Elastic Load Balancing ensure the use of newer and stronger cipher suites when establishing a new connection supporting the Server Order Preference option. When this option is selected, the load balancer selects the first cipher in its list that is in the client's list of ciphers. **Remediation:** To enable Server Order Preference: • Open the Amazon EC2 console. • Under LOAD BALANCING, choose Load Balancers. • Select your Load Balancer. • On the Listeners tab, for Cipher, choose Change. • On the Select a Cipher page, select Custom Security Policy. • For SSL Options, select Server Order Preference. • Click Save.
Use HTTPS listener for Web Tier ELB	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) 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. **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,

Configure the latest SSL Security Policies for Web Tier ELB	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) 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. **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>
Add SSL/TLS Certificate to App Tier ELB	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) All the application traffic between the Web Tier instances and the App Tier ELB nodes should be encrypted using an SSL\TLS certificate. **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,

Configure the latest SSL Security Policies for App Tier ELB	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 App Tier EC2 instances, and using a TCP listener on the ELB (TCP pass-through) 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. **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>
Use HTTPS listener for App Tier ELB	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) 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. **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,

Add SSL/TLS Certificate to Web Tier ELB	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) All the application traffic between the clients and the Web Tier ELB nodes should be encrypted using a SSL/TLS certificate. **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,
Associate each Auto-Scaling Group to ELB	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. 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: minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group Availability Zones / subnets used **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>

Set a HTTPS connection from all CloudFront Distributions to the Web Tier ELB origin	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. To ensure that objects are encrypted from edge locations to the Web-Tier ELB origin according to the data classification policy, use Match Viewer. **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>
Associate Web Tier Auto-Scaling Group to ELB	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. 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: minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group Availability Zones / subnets used **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>

Associate App Tier Auto-Scaling Group to ELB	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. 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: minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group Availability Zones / subnets used **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>

Configure Health Check for Web Tier ELB	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, attempt 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
	
	Ensure availability of back-end EC2 instances associated with an Amazon ELB through application layer health check (ex: http) instead of TCP health checks.
	
	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

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

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

Set Root Domain Alias Record to ELB	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. Route53 provides special record type called Alias that allows creation of an A record for the root domain and points 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. **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>

Allow connectivity to the VPC Internet Gateway (IGW) and associate the Routing Table with Web tier ELB subnet (by default route (0.0.0.0/0))	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. 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. **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>

Use a Web-Tier ELB Security Group to accept only HTTP/HTTPS	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 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 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.
	
	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

Do not use Web tier ELB Security Group in the Auto Scaling launch configuration of any other tier (Web, App)	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. The web-tier ELB is the only one that is public facing and should have rules to allow inbound traffic to 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. **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>

Create the App tier ELB Security Group to only accept HTTP/HTTPS	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 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.
	
	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 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

Create the App tier Security Group to allow inbound connections from App tier ELB Security Group for explicit ports	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. 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. **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

Create the App tier ELB as Internal	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. 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. **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>
Create subnets for the Web Tier ELB	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. 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. **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

Component: MySQL

Name	Description
Use prepared statements for all database queries	Database injection attacks, such as SQLi (SQL Injection) rely on sending tainted client-side data which is used in dynamic SQL queries at the server-side in an unsafe manner. For example, creating queries by concatenating strings using untrusted data may result in vulnerable code , for example an attacker may append an 'OR' statement through the provided customerName parameter passed to the following code to bypass the checks and return additional data from the database: • String query = "SELECT user FROM users WHERE name = '" + request.getParameter("customerName")+"'"; Using prepared statements with carefully controlled and validated input conditions mitigates against SQLi and related attacks. • Database queries should always be made using prepared statements or parameterized queries. • Queries through an Object-Relational mapper should also be treated as tainted input, and made using prepared statements to mitigate the threat.
Apply required security patches to the service	Vendors and other maintainers of software release patches in response to security flaws and other bugs in their products. The longer a system is exposed with a known security vulnerability, the easier to compromise it is as the exploit became public, they get included into automated exploitation suites like Metasploit and a wider audience is able to exploit them. • Apply patches and other software updates in a timely manner to prevent unexpected failures or exploitation. • Clearly define an approach for testing and applying patches, in particular security patches, with expected timescales. There is often a small window between release of a patch, and potentially malicious actors reverse-engineering the patch to identify and exploit the flaw. • Use a threat intelligence, vulnerability scanning, or other alerting service to ensure the project team is aware of issues within the project or its components promptly.
Access the data store from an account with the least privileges necessary	Use an account with only the minimum set of permissions required to access the data store. The account should not be able to perform operations that are not explicitly required by the component that performs these operations. For example, if a web application needs to read data from certain tables and insert and update data from others, then a database account with only those specific permissions should be used by the application server.
Restrict access to the service at the network layer to reduce exposure	Access to services should be restricted to expected sources, limiting exposure of the service and its attack surface; and the likelihood of a malicious actor gaining access to the system. • Apply network layer security controls so that only the necessary and expected IP addresses are permitted access to connect to the service.

Appendix A: Countermeasure Details

This appendix shows all of the countermeasures mitigating the threats found in the project.

Component: API GW

Id	Name	Description	State	Result
CWE-799	Implement application and network rate limiting	A number of attacks rely on brute-force techniques to send large volumes of requests to enumerate or attempt to exploit flaws in an application, for example, sending common passwords to multiple target accounts within an application. By profiling normal traffic volumes, and applying rate limiting, the application can be built to actively mitigate such attacks. • Connection rate-limiting based on the source IP address can be used to restrict attacks against the authentication or registration systems. Multiple failures (or attempts) from a single IP should result in temporarily blocking or dropping traffic from the source. Note however that some corporate and ISP environment may place multiple valid and discrete clients behind the same IP address, resulting in false-positives. • Attackers may use botnets and other IP masking techniques to deliver attacks from multiple sources to avoid IP based rate-limiting. To mitigate this class of attack, Indicators of Compromise should be monitored (for example a higher rate of login failures than usual), and appropriate actions taken. For example, when the application detects active brute-force attacks, a Web Application Firewall (WAF) or other intermediate devices could be used to block attacks sharing a signature from pattern matching or deep packet inspection (e.g. HTTP headers or common passwords across multiple accounts). Similarly, the application could respond by requiring a CAPTCHA, cookie, or Javascript challenge when an attack is detected. **Remediation:** Implement the mechanisms to lockout accounts: • When the application detects a set number of failure login attempts, the account shall be locked for a certain time period. • When the application detects that an account is locked more times than usual, this account shall be disabled. A disabled account shall only be restored by an administrator. • When the application detects active brute-force attacks, the application shall require a CAPTCHA, cookie, or JavaScript challenge.	Rejected	Not Tested
OTG-BUSLOGIC	Detect and notify the usage of automated tools or unusual behavior	Don't allow users to manipulate a system or guess its behavior based on input or output timing and detect the usage of automated tools or unusual behavior, such as actions not performed in reasonable "human time" or other abnormal time patterns. When the usage of automated tools is detected, the application shall respond with denying the access and notifying the security group.	Rejected	Not Tested

CWE-541	Prevent unauthorised access to source code through the service	Access to the source-code for the application can aid an attacker in determined bugs or vulnerabilities in the code or logic. For closed-source projects it is therefore important to control and restrict access to the source. Application services may unexpectedly expose code, for example a service providing files to a user could be manipulated to access source code if implemented insecurely. • Ensure that source code is not inadvertently disclosed through misconfiguration or vulnerabilities in the service. • Check that configuration files are not downloadable directly from the service, and cannot be read and viewed through the service itself. • Ensure backups, operating system, and version control artifacts do not expose code.	Recommended	Not Tested
CDS-USER-TRACK	Log details of user actions within the system	To maintain proper accountability, logs should be maintained with sufficient information to track user actions within the system. These logs should be forensically sound, non-repudiable, and contain comprehensive details about activity. While the exact data for an event may vary, the following should be captured at a minimum: • Timestamps against a proven external source (e.g. an NTP server) • Origin, with this field we mark if the logs are provided by a trusted or untrusted source. • Event, status, and/or error codes (with sensitive data masked as appropriate or not introduced in logs) • Service, command, application or function name and details • User or system account associated with an event • Devices used (e.g. source and destination IPs, terminal session ID, web browser, etc) Source: https://security.berkeley.edu/security-audit-logging-guideline	Required	Not Tested
CSD-VAL-LOG	Log and reject all data validation failures	Data validation failures, together with access control violations, are symptomatic of malicious activity where an attacker is attempting to subvert the protections in place. It is therefore likely that unexpected input detected by the application relates to an attack. Rejecting and logging such activity, and ideally terminating the session, increases the likelihood of detecting and inhibiting structured attacks against the application. • Log all validation failures when rejecting requests. • Ensure logged data is appropriately sanitized and encoded to prevent attacks against the logs and subsequent access to them. • Terminate the offending user session to inhibit further attack. • Ensure errors returned to the client-side are generic to prevent an attacker enumerating the defenses in place or gaining knowledge about the back-end.	Required	Not Tested

CWE-319-TRANSPORT	Encrypt data between the client and server/service	Data passed between the client and server should be protected by encryption in transit. • Implement cryptographically strong TLS end-to-end encryption between the client and server, terminating within a secure environment on the server-side. • Consider use of client certificates to prevent interception of (or man-in-the-middle attacks on) the encrypted connection. • Alternatively, asymmetric (public-key) encryption could be utilized, although a recognized, proven, and tested implementation/library should be used	Required	Not Tested
CWE-662	Use a synchronised time source	In order to correlate logs and data from different internal and external systems, and to preserve forensic quality of the logs, it is important a unified and trusted synchronized time source is used throughout the environment. • Servers should be synchronize to an internal or external NTP server • The centralized source should in turn use (or be) a trusted central time source. This control is critical in identifying application events (including attacks) through logging, and in conduction post-event analysis, in particular to track the whole user (or attacker) journey through the system should it be compromised. It is good practice to use the concept of Indicators of Compromise (IoC) should be used to detect possible situations in which the system has been compromised and to give an appropriate response. IoCs are often tracked through logs, and accurate time is often essential.	Required	Not Tested

DATA-VAL	Validate all data received from the client side	All data received from the client-side should be considered tainted and a potential risk, regardless of the source or transport method. For example, while hidden form fields, cookies, or other headers may be obfuscated from a user, along with parameters passed in ViewStates or other encapsulated forms, these can be modified by the user at the client-side in memory, or in transit on the network. Similarly, data passed from binary or compiled components can be modified in situ or in transit. *Furthermore, encryption only secures the data in transit between the two ends of the encrypted tunnel (one end of which is typically controlled by the client); data passing through the link may still be malicious.* As such, all data from the client side must be subjected to strict validation, sanitization, and encoding against expected syntactic and semantic criteria. • Define a specification of the data that is expected at each input; both the syntax (e.g. alphanumeric only) and semantics (e.g. a word of between 1 and 25 characters, or a specific list). As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if the input is only expected to contain colors such as "red" or "blue." • Implement a 'known good' or white-list approach, where only inputs that meet the strict criteria for each input are accepted, and reject, transform, or encapsulate any non-compliant data. • While useful for identifying malicious content, do not rely on looking for specific malformed or attack payloads (blacklists). It is almost impossible to maintain a comprehensive and accurate blacklist due to the complexity and evolving nature of attacks, opportunities to obfuscate payloads, and changes to the code's execution environment. As noted, blacklists can be useful for detecting and logging potential attacks, or determining which inputs are so malformed that they should be rejected outright. • Validate all data received from the client, including values such as HTTP headers and cookie values if these are used as input on the server side, X- headers, and other platform specific data objects passed between the client and server.	Required	Not Tested
LOG-TLS-FAILURES	Log the backend TLS connection failures	Implement functionality to record backend TLS connection failures and include these in the logs.	Required	Not Tested
LOGS-INTEGRITY	Ensure the integrity of the logging system	Ensure Log integrity for the application generated logs, such as storing logs on write-once media, forwarding a copy of the logs to a centralized SIEM or generating message digests for each log file.	Required	Not Tested
RESTRICT-NUMBER-ACCOUNT-TO-LOGS	Limit the number of accounts with privileges allowing modification and/or deletion of audit logs files	Limit the number of account with privileges to modify and/or delete audit logs files.	Required	Failed
SAME-ENCODING-STYLE	Ensure that the client-side and the server-side are using the same encoding style	Ensure that the client-side and the server-side are using the same encoding style.	Required	Not Tested

ASVS-8.8	Escape meta-characters from un-trusted data	If untrusted data, including any data received from the client side of a connection is directly written to a log file, then this data could contain newline or other meta-characters that would allow an attacker to forge log entries. Such meta-characters should first be escaped or removed before the data is written to the logging system.	Implemented	Passed
CWE-532	Do not write secrets to the log files	The logs may be accessed by attackers and in order to protect sensitive data, no such sensitive data should be included in the logs	Implemented	Not Tested
LOG-RETENTION	Develop a log retention policy	Develop a log retention policy to identify storage requirements for device logs and implement procedures to ensure that the audit logs are available for a security response in the case of incident or investigation. The audit logs must be collected for the last 30 days in easily accessible storage media. Older logs should be archived in a protected storage and should be accessible in the future as required for incidents or investigations.	Implemented	Not Tested
OTG-BUSLOGIC-006	Restrict actions of users that follow unusual patterns.	Restrict actions that users can do outside of the approved/required business process flow. This is important because without this safeguard in place attackers may be able to bypass or circumvent work-flows and checks allowing them to prematurely enter or skip required sections of the application potentially allowing action/transaction to be completed without successfully completing the entire business process, leaving the system with incomplete back-end tracking information.	Implemented	Not Tested

Component: ELB - Elastic Load Balancer

Id	Name	Description	State	Result
Hydras-AWS-ELB-01	Use the Perfect Forward Secrecy feature	For greater communication privacy Elastic Load Balancing allows the use of Perfect Forward Secrecy. This feature provides additional safeguards against eavesdropping on encrypted data, through the use of a unique random session key, and therefore prevents the decoding of captured data, even if the secret long-term key is compromised. **To begin using Perfect Forward Secrecy:** Configure your load balancer with the newly added Elliptic Curve Cryptography (ECDHE) cipher suites.	Recommended	Not Tested
Hydras-AWS-ELB-02	Select the Server Order Preference option	Within Elastic Load Balancing ensure the use of newer and stronger cipher suites when establishing a new connection supporting the Server Order Preference option. When this option is selected, the load balancer selects the first cipher in its list that is in the client's list of ciphers. **Remediation:** To enable Server Order Preference: • Open the Amazon EC2 console. • Under LOAD BALANCING, choose Load Balancers. • Select your Load Balancer. • On the Listeners tab, for Cipher, choose Change. • On the Select a Cipher page, select Custom Security Policy. • For SSL Options, select Server Order Preference. • Click Save.	Recommended	Not Tested

aws-tier-1.1	Use HTTPS listener for Web Tier ELB	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) 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. **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	Recommended	Not Tested

aws-tier-1.10	Configure the latest SSL Security Policies for Web Tier ELB	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) 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. **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>	Recommended	Not Tested

aws-tier-1.12	Add SSL/TLS Certificate to App Tier ELB	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) All the application traffic between the Web Tier instances and the App Tier ELB nodes should be encrypted using an SSL\TLS certificate. **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	Recommended	Not Tested

aws-tier-1.13	Configure the latest SSL Security Policies for App Tier ELB	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 App Tier EC2 instances, and using a TCP listener on the ELB (TCP pass-through) 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. **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>	Recommended	Not Tested

aws-tier-1.14	Use HTTPS listener for App Tier ELB	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) 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. **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	Recommended	Not Tested

aws-tier-1.9	Add SSL/TLS Certificate to Web Tier ELB	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) All the application traffic between the clients and the Web Tier ELB nodes should be encrypted using a SSL/TLS certificate. **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	Recommended	Not Tested

aws-tier-3.1	Associate each Auto-Scaling Group to ELB	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. 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: minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group Availability Zones / subnets used **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>	Recommended	Not Tested

aws-tier-3.12	Set a HTTPS connection from all CloudFront Distributions to the Web Tier ELB origin	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. To ensure that objects are encrypted from edge locations to the Web-Tier ELB origin according to the data classification policy, use Match Viewer. **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>	Recommended	Not Tested

aws-tier-3.13	Associate Web Tier Auto-Scaling Group to ELB	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. 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: minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group Availability Zones / subnets used **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>	Recommended	Not Tested

aws-tier-3.14	Associate App Tier Auto-Scaling Group to ELB	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. 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: minimum / maximum number of EC2 instances to be launched by the Auto-Scaling Group Availability Zones / subnets used **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>	Recommended	Not Tested

aws-tier-3.8	Configure Health Check for Web Tier ELB	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, attempt 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
		
		Ensure availability of back-end EC2 instances associated with an Amazon ELB through application layer health check (ex: http) instead of TCP health checks.
		
		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
			Recommended	Not Tested

aws-tier-3.9	Configure Health Check for App Tier ELB	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
		
		Ensures availability of back-end EC2 instances associated with an Amazon ELB through application layer health check (ex: http) instead of TCP health checks.
		
		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
			Recommended	Not Tested

aws-tier-5.2	Enable the ELB logging	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.
		
		AWS Elastic Load Balancers (ELBs) can record all incoming requests sent to the load balancer and store within logs on S3. This allows for diagnosing application failures and analyzing web traffic and security analysis of incoming traffic
		
		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
			Recommended	Not Tested

aws-tier-6.1	Set Root Domain Alias Record to ELB	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. Route53 provides special record type called Alias that allows creation of an A record for the root domain and points 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. **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>	Recommended	Not Tested

aws-tier-6.13	Allow connectivity to the VPC Internet Gateway (IGW) and associate the Routing Table with Web tier ELB subnet (by default route (0.0.0.0/0))	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. 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. **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>	Recommended	Not Tested

aws-tier-6.17	Use a Web-Tier ELB Security Group to accept only HTTP/HTTPS	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 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 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.
		
		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
			Recommended	Not Tested

aws-tier-6.18	Do not use Web tier ELB Security Group in the Auto Scaling launch configuration of any other tier (Web, App)	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. The web-tier ELB is the only one that is public facing and should have rules to allow inbound traffic to 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. **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>	Recommended	Not Tested

aws-tier-6.21	Create the App tier ELB Security Group to only accept HTTP/HTTPS	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 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.
		
		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 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":
			Recommended	Not Tested

		"<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		
aws-tier-6.22	Create the App tier Security Group to allow inbound connections from App tier ELB Security Group for explicit ports	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. 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. **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	Recommended	Not Tested

aws-tier-6.26	Create the App tier ELB as Internal	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. 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. **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>	Recommended	Not Tested

aws-tier-6.5	Create subnets for the Web Tier ELB	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. 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. **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	Recommended	Not Tested

Component: MySQL

Id	Name	Description	State	Result
RESTRICT-SERVICE	Restrict access to the service at the network layer to reduce exposure	Access to services should be restricted to expected sources, limiting exposure of the service and its attack surface; and the likelihood of a malicious actor gaining access to the system. • Apply network layer security controls so that only the necessary and expected IP addresses are permitted access to connect to the service.	Recommended	Not Tested
CWE-89-PREPARED	Use prepared statements for all database queries	Database injection attacks, such as SQLi (SQL Injection) rely on sending tainted client-side data which is used in dynamic SQL queries at the server-side in an unsafe manner. For example, creating queries by concatenating strings using untrusted data may result in vulnerable code , for example an attacker may append an 'OR' statement through the provided customerName parameter passed to the following code to bypass the checks and return additional data from the database: • String query = "SELECT user FROM users WHERE name = '" + request.getParameter("customerName")+"'"; Using prepared statements with carefully controlled and validated input conditions mitigates against SQLi and related attacks. • Database queries should always be made using prepared statements or parameterized queries. • Queries through an Object-Relational mapper should also be treated as tainted input, and made using prepared statements to mitigate the threat.	Required	Not Tested
PATCH-SERVICE	Apply required security patches to the service	Vendors and other maintainers of software release patches in response to security flaws and other bugs in their products. The longer a system is exposed with a known security vulnerability, the easier to compromise it is as the exploit became public, they get included into automated exploitation suites like Metasploit and a wider audience is able to exploit them. • Apply patches and other software updates in a timely manner to prevent unexpected failures or exploitation. • Clearly define an approach for testing and applying patches, in particular security patches, with expected timescales. There is often a small window between release of a patch, and potentially malicious actors reverse-engineering the patch to identify and exploit the flaw. • Use a threat intelligence, vulnerability scanning, or other alerting service to ensure the project team is aware of issues within the project or its components promptly.	Required	Not Tested

CWE-306-SERVICE	Require authentication before presenting restricted data	The application should ensure users have undergone an Identification and Verification (ID&V) process before allowing access to secret, sensitive or otherwise restricted data. For less sensitive but still restricted data, simple verification of the location of the user may suffice (e.g. IP restrictions). • For non-sensitive but non-public data, access could be restricted by IP address, for example limiting access to internal networks, workstations, or gateways • For more sensitive data, TLS client-side certificates may be appropriate • Where secret or other sensitive data is handled, a full authentication process to identify and validate users with single or multi-factor authentication may be required	Implemented	Failed
RESTRICT-ACCESS-DATABASE	Access the data store from an account with the least privileges necessary	Use an account with only the minimum set of permissions required to access the data store. The account should not be able to perform operations that are not explicitly required by the component that performs these operations. For example, if a web application needs to read data from certain tables and insert and update data from others, then a database account with only those specific permissions should be used by the application server.	Implemented	Not Tested