AWS EC2 Security Checks

Category	Check Name	Command	Explanation
Kafka Core	Broker Port Exposure	nmap -p 9092 <broker-ip>	Ensure the Kafka broker port (default 9092) is not publicly accessible. It should be restricted via firewalls or security groups to known sources.
Kafka Core	Zookeeper Port Exposure	nmap -p 2181 <zookeeper-ip>	Zookeeper should not be exposed to the public. Access should be limited to Kafka brokers and trusted admin hosts.
Kafka Core	Unencrypted Communication	grep -i 'PLAINTEXT' server.properties	Look for plaintext protocols in the config file server.properties. Use TLS encryption (SSL) instead of PLAINTEXT to secure data in transit.
Kafka Core	Missing Authentication	grep -i 'sasl.enabled.mechanisms' server.properties	Check if SASL authentication is enabled. Without authentication, any client can connect and interact with the cluster. **sasl.enabled.mechanisms=PLAIN, GSSAPI and JAAS files configured**
Kafka Core	Weak ACL Configurations	kafka-acls.sh --list --bootstrap-server <broker>:9092	Ensure ACLs (Access Control Lists) are applied to all topics, consumer groups, and transactions to prevent unauthorized access.
Kafka Core	Zookeeper ACLs	zookeeper-shell.sh <zookeeper>:2181 getAcl /	Check if Zookeeper ACLs are set. Default settings may allow unauthenticated access. **Zookeeper allows world access (auth: world:anyone)**
Kafka Core	Inter-Broker Encryption	grep -i 'security.inter.broker.protocol' server.properties grep -i 'security.inter.broker.protocol' kafka.properties	Verify that inter-broker communication uses SSL instead of PLAINTEXT for encryption and integrity. security.inter.broker.protocol=SSL / SASL_SSL security.inter.broker.protocol=PLAINTEXT
Kafka Core	JMX (Java Management Extensions) Remote Port Exposure Download jmxterm.jar https://repo1.maven.org/maven2/org/cyclopsgroup/jmxterm/1.0.1/jmxterm-1.0.1-uber.jar	nmap -p 9999 <broker-ip> ps aux | grep kafka **Check for custom ports which are running java:** env | grep JMX sudo ss -tulp | grep java sudo netstat -tulp | grep java	You are looking for: -Dcom.sun.management.jmxremote=true -Dcom.sun.management.jmxremote.port=9999 (default port) -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.ssl=false Check if JMX ports are exposed. These should be firewalled or tunneled securely, as they can expose sensitive metrics and controls. If JMX is exposed outside the local machine, try to connect to it: **java -jar jmxterm.jar -l <machine-a-ip>:9999** **If you get a prompt like:** $> (it means you are in unauthenticated) **List MBeans** $> beans #domain = kafka.log:

Kafka Core	Unsecured REST Proxy	(Look for **-Drest.port=8082**) ps aux | grep kafka-rest curl http://<rest-proxy>:8082/topics curl https://<rest-proxy>:8082/topics -k **Check for custom ports which are running java:** sudo ss -tulp | grep java sudo netstat -tulp | grep java	Verify if the REST proxy is not **publicly accessible** and is **secured (TLS)** using authentication and authorization. Check kafka-rest.properties for: listeners=http://0.0.0.0:8082 listeners=https://0.0.0.0:8443 (SSL) ssl.keystore.location=/../../kafka-rest.keystore.jks ...
Kafka Core	Server.properties - Listener Protocols	cat /path/to/kafka/config/server.properties | grep listeners	Verify the broker is not using PLAINTEXT protocol. Look for 'SSL' or 'SASL_SSL' in the listeners config. listeners=**SASL_SSL**://broker:9093 listeners=**PLAINTEXT**://0.0.0.0:9092
Kafka Core	Server.properties - Inter-broker Protocol	cat /path/to/kafka/config/server.properties cat /path/to/kafka/config/kafka.properties	Ensure that brokers communicate using secure protocol (SSL or SASL_SSL). security.inter.broker.protocol=**SASL_SSL** inter.broker.listener.name=**SASL_SSL** security.inter.broker.protocol=**PLAINTEXT**

Kafka Core	JAAS Config - Hardcoded or Weak Credentials	cat /path/to/kafka/config/kafka_server_jaas.conf	Check for presence of hardcoded or default usernames/passwords in server.properties or JAAS config files.
			KafkaServer {
			org.apache.kafka.common.security.**scram.ScramLoginModule** required
			username="admin"
			password="long-random-secret";
			};
			
			KafkaServer {
			org.apache.kafka.common.security.**plain.PlainLoginModule** required
			username="admin"
			password="**admin**";
			};

Kafka Core	Open Kafka Connectors	**Check is kafka connect is running:** ps aux | grep connect **Look for running proccesses that include the line:** org.apache.kafka.connect.cli.ConnectDistributed /path/kafka-connect.properties **Try to reach the connector REST API endpoint unauthenticated:** curl http://<connect-host>:8083/connectors	Ensure connectors are not accessible without authentication. An open connector endpoint can be misused to exfiltrate data. **Connect REST API authenticated via BasicAuth or OAuth**

Kafka Core	Unauthenticated Kafka Broker Access	kafka-topics.sh --bootstrap-server <broker>:9092 --list	Check if a client can list topics without authentication. If successful, the broker is accepting unauthenticated access.
Kafka Core	Unauthenticated Topic Creation	kafka-topics.sh --create --bootstrap-server <broker>:9092 --topic test-intrusion	Attempt to create a topic as an unauthenticated user. Success indicates missing ACLs or authentication controls. ACLs restrict topic creation to admin users only, using SASL credentials.
Kafka Core	Create Topic as Unauthorized User	kafka-topics.sh --bootstrap-server <broker>:9092 --create --topic pentest-topic --partitions 1 --replication-factor 1	Check if topic creation is allowed without proper authorization. If this succeeds, ACLs are likely misconfigured.
Kafka Core	Alter Topic Configuration without Auth	kafka-configs.sh --bootstrap-server <broker>:9092 --alter --entity-type topics --entity-name test-topic --add-config retention.ms=100000	Attempt to change topic settings. This should require authentication and authorization.
Kafka Core	SSL Certificate Validation	openssl s_client -connect <broker>:9093	Verify SSL certificates are valid, not expired, and issued by a trusted CA. Check for correct hostname binding. Valid, signed SSL cert with matching CN/SAN Self-signed, expired SSL cert used
Zookeeper	Unauthenticated Zookeeper Access. List Kafka Brokers from Zookeeper	**Get the zookeeper address & port** cat /path/kafka/server.properties zookeeper.connect=test1.test.local:2181 echo 'ls /brokers/ids' | zookeeper-shell.sh <zookeeper>:2181 ./zookeeper-shell.sh <zookeeper>:2181 ls /brokers/ids	Zookeeper should restrict access to broker metadata. This command should not succeed unauthenticated.
Zookeeper	Get Topic Config via Zookeeper	echo 'get /config/topics/<topic>' | zookeeper-shell.sh <zookeeper>:2181	Shows if topic metadata and configs are exposed directly in Zookeeper. Should be protected.
Zookeeper	Zookeeper Node Write Test	echo 'create /test_znode "hack"' | zookeeper-shell.sh <zookeeper>:2181	Test whether a new node can be created in Zookeeper without proper ACLs.
Zookeeper	Zookeeper World ACL Misconfiguration	zookeeper-shell.sh <zookeeper>:2181 getAcl /	World ACL grants full access to anyone, exposing sensitive config or allowing unauthorized changes. Only authenticated users have read/write access, **world:anyone** removed. ACL: '**auth:anyone:cdrwa**' allows complete control to unauthenticated users.
Zookeeper	Zookeeper Configuration - No Authentication	grep 'authProvider' /path/to/zookeeper/conf/zoo.cfg cat /path/kafka/server.properties	Ensure Zookeeper is set to require authentication (e.g., SASL or digest). authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider #authProvider is not set
Kafka Streams	Streams Internal Topics ACLs	kafka-acls.sh --list --bootstrap-server <broker>:9092	Kafka Streams creates internal topics (e.g., repartition, changelog). Ensure ACLs protect these topics from external consumers or producers. Only SSL listeners are enabled in server.properties PLAINTEXT listeners enabled

Kafka Streams	Streams Metrics Exposure	curl http://<metrics-endpoint>:<port>/metrics	If using JMX or Prometheus exporters for Kafka Streams, ensure the endpoint is secured and not exposed to the public to prevent data leakage or
Kafka Connect	Inspect all defined ACLs in the Kafka Cluster	kafka-acls.sh --authorizer-properties zookeeper.connect=<zookeeper>:2181 --list	Current ACLs for resource `Topic:test-topic`: User:CN=producer has Allow permission for operations: Write from hosts: 10.0.0.1 User:* has Allow permission for operations: All from hosts: * If the command **returns no ACLs at all**, it likely means ACLs are not enabled. This is a critical issue — anyone can access Kafka without restriction unless network security compensates.
Jolokia Agent	Jolokia Agent Exposure	curl http://<host>:8778/jolokia	Jolokia exposes JMX data over HTTP. Ensure it is not publicly accessible and has authentication and IP whitelisting enabled.
Jolokia Agent	Jolokia Weak Access Control	curl http://<host>:8778/jolokia/read/java.lang:type=Memory	Test whether sensitive MBeans are accessible without authentication. Jolokia should be configured to restrict access to specific MBeans.
Schema Registry	Schema Registry API Exposure	Check configuration files such as schema-registry.properties or server.properties: **SCHEMA_REGISTRY_LISTENERS**: http://0.0.0.0:8081 **listener**s=http://0.0.0.0:8081 curl http://<schema-registry-host>:8081/subjects	Ensure the Schema Registry API is not exposed publicly and is protected by authentication and network-level controls.
Schema Registry	Schema Registry Config - SSL Disabled	grep 'listeners' /path/to/schema-registry/schema-registry.properties	Ensure only HTTPS (SSL) is enabled for the schema registry endpoint. listeners=https://0.0.0.0:8081 listeners=http://0.0.0.0:8081