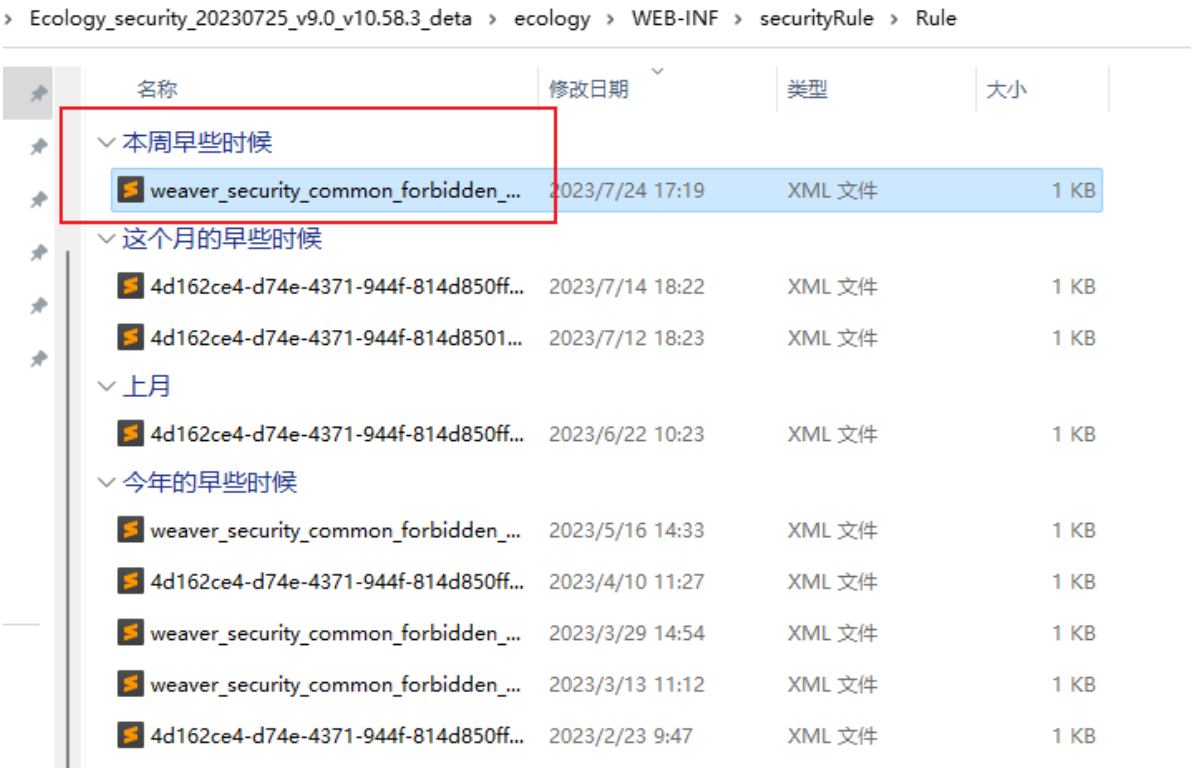
泛微OA_E-Cology 前台节点升级文件上传漏洞

简单看了一下，发现该漏洞比较鸡肋，在此记录一下

在泛微的官网上下载 10.58.3的补丁在里面翻了一下，



噢，发现新增了几条规则，那看来问题应该就是出自这里了

```
<?xml version="1.0" encoding="utf-8"?>

<root>
 <forbidden>
        <need-check-urls>
            <url>/api/import/export/check</url>
            <url>/clusterupgrade/clusterNodeOperation.jsp</url>
            <url>/clusterupgrade/clusterSettingOperation.jsp</url>
            <url>/clusterupgrade/clusterUpgrade.jsp</url>
            <url>/clusterupgrade/uploadFileClient.jsp</url>
            <url>/clusterupgrade/uploadFileServer.jsp</url>
        </need-check-urls>
    </forbidden>
</root>
```

从下往上逐个看，先看 uploadFileServer.jsp 发现要用户身份，所以先放一放

```jsp
uploadFileServer.jsp  ClientOperation.class  uploadFileClient.jsp  checkUpgradeStatus.jsp
<%@ page language="java" contentType="application/json; charset=utf-8"%>
<%@ page import="java.util.*"%>
<%@ page import="java.text.*"%>
<%@ page import="weaver.upgradetool.upgrade.ClusterOperation"%>
<%@ page import="weaver.hrm.User" %>
<%@ page import="weaver.hrm.HrmUserVarify" %>
<%

    User user = HrmUserVarify.getUser (request , response) ;
    if(user == null)  return ;

    ClusterOperation cluster = new ClusterOperation();
    String res = cluster.uploadFileServer(request);
    out.print(res);
%>
```

再来看看 uploadFileClient.jsp 此处获取了个token校验了一下就进行文件上传了，好像有戏

```jsp
uploadFileServer.jsp  ClientOperation.class  uploadFileClient.jsp  checkUpgradeStatus.jsp
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
<%@ page language="java" contentType="application/json; charset=utf-8"%>
<%@ page import="java.util.*"%>
<%@ page import="java.io.*"%>
<%@ page import="net.sf.json.JSONObject"%>
<%@ page import="weaver.upgradetool.upgrade.UploadHandle"%>
<%@ page import="weaver.upgradetool.upgrade.ClientOperation"%>
<%
    response.setCharacterEncoding("UTF-8");

    String token = (String) request.getParameter("token");
    ClientOperation cluster = new ClientOperation();

    String message = "";
    boolean checkSecurity = cluster.checkSecurity(token, request);
    if(checkSecurity) {
        message= "\u\u\u\u\u\u\u\u\u";
    }
    message = new UploadHandle().upload(request);
    out.print(message);
%>
```

跟进upload函数，一开始我也以为是这个地方出现了问题，然而发现这里只能上传zip

这里只能算作是个上传的接口（这里需要吐槽的是，泛微的代码实在太混乱了，居然还有另一个同名同方法的包，两个包分别在`ecology\web\resin_lib\lib\clusterupgrade.jar`和`ecology\web\WEB-INF\lib\clusterupgrade.jar`）这里服务器会加载的是WEB-INF\lib 下面的jar包，然而发现代码中只能上传zip的压缩包。

```
if (!var20.endsWith("zip")) {
    var5 = "error,补丁包格式不规范，无法升级！";
    String var21 = var5;
    return var21;
}
```

于是再去看其他的接口

在 `ecology\web\clusterupgrade\clusterUpgrade.jsp` 中终于看到了关键的地方

在代码中，接收了一个method，一个token，根据method可以进入上传的分支

```
<%@ page language="java" contentType="application/json; charset=utf-8"%>
<%@ page import="java.util.*"%>
<%@ page import="java.io.*"%>
<%@ page import="weaver.upgradetool.upgrade.ClientOperation"%>
<%
    response.setCharacterEncoding("UTF-8");
    String method = request.getParameter("method");
    ClientOperation cluster = new ClientOperation();
    String token = (String) request.getParameter("token");
    //System.out.println("token:"+token);

    if("getinfo".equals(method)) {
        HashMap<String,String> param = new HashMap<~>();
        param.put("token",token);
        String msg = cluster.getUpgradeProcess(param,request);
        out.print(msg);
    } else if("upgrade".equals(method)){
            boolean checkSecurity = cluster.checkSecurity(token, request);
            if(!checkSecurity) {
                return;
            }
            //System.out.println("====method:"+method);
            HashMap<String,String> param = new HashMap<~>();
            cluster.upgrade(param,request);
    } else {//上传
            boolean checkSecurity = cluster.checkSecurity(token, request);
            if(!checkSecurity) {
                return;
            }
            cluster.uploadFileClient(request);
    }
```

那么我们呢是否可以让checkSecurity返回为true呢？ 进入checkSecurity方法， 让var4变为true就得先让var3变成true，才能进去checkSecurity，所以我们的进入checkIp方法看看

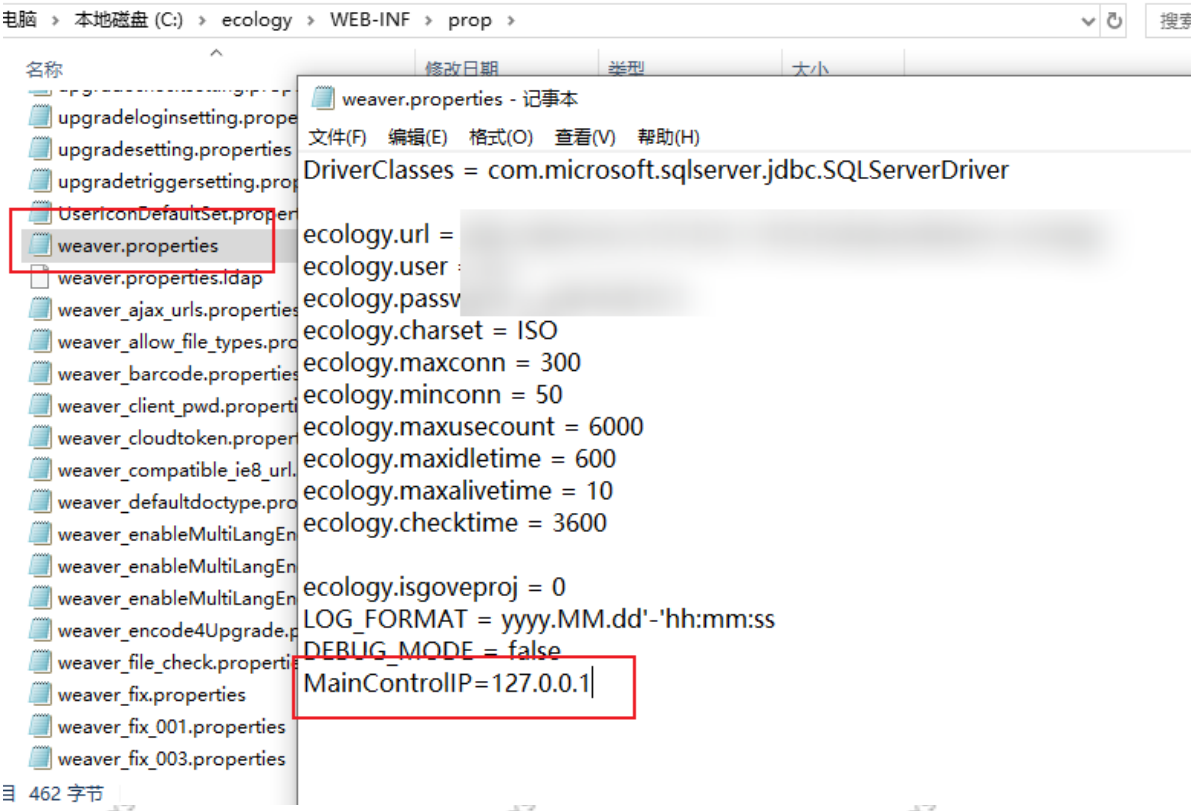
```java
public boolean checkSecurity(String var1, HttpServletRequest var2) {
    boolean var3 = this.checkIp(var2);
    boolean var4 = false;
    if (var3) {
        var4 = this.checkSecurity(var1);
    } else {
        var4 = false;
    }

    if (!var4) {
        ClusterUpgrade.getInstance().setErrorMsg("error,验证不通过（请使用内网地址访问系统升级）");
    }

    return var4;
}
```

此处，也是让这个漏洞大大削弱了可利用价值的地方，此处从泛微的配置文件中获取了一个MainControlIP的值与http获取的remoteIP进行对比，只有相同时返回的才是true。

而且这个MainControlIP只有在集群部署的时候才会设置，而且还得猜这个值是什么，目标可并不一定是会设置 127.0.0.1的。



不过还好，在内网时获取remoteIP时可以通过X-Forwarded-For来伪造IP，那么还是有一定几率可以打成功的。

```java
public boolean checkIp(HttpServletRequest var1) {
    CategoryUtil var2 = new CategoryUtil();
    String var3 = var2.getCategory();
    if (var3.equals("ecology")) {
        String var4 = this.getIp(var1);
        OrderProperties var5 = new OrderProperties();
        var5.load( s: GCONSTUClient.getPropertyPath() + "weaver.properties");
        String var6 = Util.null2String(var5.get("MainControlIP"));
        this.loger.info( o: "clusterupgrade================mainControlIp:" + var6);
        this.loger.info( o: "clusterupgrade================ip:" + var4);
        if (var4 == null) {
            return true;
        } else {
            return !"".equals(var6) && var6.equals(var4);
        }
    } else {
        return true;
    }
}
```

继续看回代码，需要进去checkSecurity方法来判断true 还是false， 此处通过反射的方法调用了check方法

```java
public boolean checkSecurity(String var1) {
    boolean var2 = this.clearSqlCache();
    this.loger.info( o: "清空sql缓存,防止出现sql不同步导致的校验不通过,清理结果:" + var2);
    CategoryUtil var3 = new CategoryUtil();
    String var4 = var3.getCategory();
    if (var4.equals("ecology")) {
        try {
            Class var5 = Class.forName("weaver.upgradetool.upgrade.CheckSecurity");
            Method var6 = var5.getMethod( name: "check", String.class);
            Object var7 = var6.invoke(var5.newInstance(), var1);
            boolean var8 = Boolean.parseBoolean(var7.toString());
            return var8;
        } catch (Exception var9) {
            var9.printStackTrace();
            return true;
        }
    } else {
        return var4.equals("emobile") ? true : true;
    }
}
```

通过 传入的token 和 以 license作为加密密钥 对 wEAver2018 +对应的时间 进行加密对比，相同则是true

```java
public boolean check(String var1) {
    RecordSet var2 = new RecordSet();
    var2.executeQuery( s: "select license from license", new Object[0]);
    String var3 = "";
    if (var2.next()) {
        var3 = var2.getString( s: "license");
    }

    try {
        DesUtils var4 = new DesUtils(var3);
        String var5 = var4.getDistributeinfo();
        String var6 = "wEAver2018" + var5;
        if (var5 == null || "".equals(var5)) {
            return false;
        }

        String var7 = var4.encrypt(var6);
        if (var1.equals(var7)) {
            return true;
        }
    } catch (Exception var8) {
        var8.printStackTrace();
    }

    return false;
}
```

那么要是我们知道密钥 （即 license），我们就能伪造加密的字符串了

那么问题来了，怎么才能获取到 license呢，在同级目录下的 tokenCheck.jsp 中，返回了我们想要的东西

我们知道它加密的密钥，即可解密出 license 是什么

```
<%
String token = request.getParameter("token");
if(token==null||token.equals("")){
    out.print("{\"status\":\"failed\"}");
}
DesUtils des = new DesUtils();
String localtoken = des.getDistributeinfo();
String timestamp = des.getDistributeinfo();


RecordSet rs = new RecordSet();
rs.executeSql("select license from license");
String key  = "";
if(rs.next()) {
    key = rs.getString("license");
}
DesUtils d = new DesUtils("ecology2018_upgrade");
key = d.encrypt(key);
out.print("{\"timestamp\":\""+timestamp+"\",\"key\":\""+key+"\"}");
return ;
/* out.print("{\"status\":\"success\"}");
%>
```

返回到 clusterUpgrade.jsp 至此，我们成功让checkSecurity变成了true，终于可以进入心心念念的uploadFileClient方法

```
        cluster.upgrade(param,request);
} else {//上传
        boolean checkSecurity = cluster.checkSecurity(token, request);
        if(!checkSecurity) {
            return;
        }
        cluster.uploadFileClient(request);
}
```

此处与 uploadFileClient.jsp 类似，也用了 upload的方法取上传zip

```java
public boolean uploadFileClient(HttpServletRequest var1) {
    try {
        ClusterUpgrade.getInstance().setUpgradeProcess(1);
        String var2 = (new UploadHandle()).upload(var1);
        if (!"".equals(var2)) {
            ClusterUpgrade.getInstance().setUpgradeProcess(0);
            ClusterUpgrade.getInstance().setErrorMsg(var2);
        }
    } catch (Exception var3) {
        var3.printStackTrace();
    }

    return true;
}
```

同样也是只能上传zip

```java
DiskFileItemFactory var14 = new DiskFileItemFactory();
var14.setSizeThreshold(102400);
var14.setRepository(var11);
ServletFileUpload var34 = new ServletFileUpload(var14);
var34.setHeaderEncoding("UTF-8");
var34.setSizeMax(-2147483648L);
List var16 = var34.parseRequest(var1);
Iterator var17 = var16.iterator();

while(var17.hasNext()) {
    FileItem var18 = (FileItem)var17.next();
    String var19;
    String var20;
    if (var18.isFormField()) {
        var19 = var18.getFieldName();
        var20 = var18.getString("UTF-8");
    } else {
        var19 = var18.getFieldName();
        var20 = var18.getName();
        if (var20 != null && !var20.trim().equals("")) {
            var20 = var20.substring(var20.lastIndexOf(File.separatorChar) + 1);
            if (!var20.endsWith("zip")) {
                var5 = "error,补丁包格式不规范，无法升级！";
                String var21 = var5;
                return var21;
            }

            var20.substring(var20.lastIndexOf( str: ".") + 1);
            var12 = var18.getInputStream();
            var13 = new FileOutputStream( name: var7 + File.separatorChar + var20);
            byte[] var24 = new byte[1024];
            boolean var25 = false;

            int var35;
            while((var35 = var12.read(var24)) > 0) {
                var13.write(var24,  off: 0, var35);
            }
```

成功上传后会存放压缩包到 `ecology\WEB-INF\versionupgrade\upload` 这个目录下

虽然只能上传压缩包，但是没关系，既然要升级，就要解压文件出来，

回到clusterUpgrade.jsp，里面还有一个method=upgrade的分支，这个地方也是我们可以控制进入的分支

```
<%
    response.setCharacterEncoding("UTF-8");
    String method = request.getParameter("method");
    ClientOperation cluster = new ClientOperation();
    String token = (String) request.getParameter("token");
    //System.out.println("token:"+token);

    if("getinfo".equals(method)) {
        HashMap<String,String> param = new HashMap<~>();
        param.put("token",token);
        String msg = cluster.getUpgradeProcess(param,request);
        out.print(msg);
    } else if("upgrade".equals(method)){
            boolean checkSecurity = cluster.checkSecurity(token, request);
            if(!checkSecurity) {
                return;
            }
            //System.out.println("====method:"+method);
            HashMap<String,String> param = new HashMap<~>();
            cluster.upgrade(param,request);
    } else {//上传
            boolean checkSecurity = cluster.checkSecurity(token, request);
            if(!checkSecurity) {
                return;
            }
            cluster.uploadFileClient(request);
    }
```

进入upgrade方法，启动了一个线程进行升级

```java
public void run() {
    String var1 = this.request.getParameter( s: "token");
    ClientOperation var2 = new ClientOperation();
    String var3 = "";
    boolean var4 = var2.checkSecurity(var1, this.request);
    if (var4) {
        ClusterUpgrade var5 = ClusterUpgrade.getInstance();

        try {
            HashMap var6 = var5.unZip(this.request, (HttpServletResponse)null);
            String var7 = (String)var6.get("iscanupdate");
            String var8;
            if ("true".equals(var7)) {
                var8 = var5.checkProcess(this.request, (HttpServletResponse)null);
                if ("canUpgrade".equals(var8)) {
                    String var9 = var5.update(this.request, (HttpServletResponse)null);
                    if (var9 == null) {
                        var5.setUpgradeProcess(5);
                    } else {
                        var5.setErrorMsg("error," + var9);
                    }
                } else {
                    var5.setErrorMsg("error," + var8);
                }
            } else {
                var8 = (String)var6.get("canupdatemessage");
                ClusterUpgrade.getInstance().setUpgradeProcess(0);
                var5.setErrorMsg("error," + var8);
            }
        } catch (Exception var10) {
            var10.printStackTrace();
        }
    }
```

var5 因为是getAttribute根本就获取不到这个值，所以默认进入else分支，此处会获取 ecology\WEB-INF\versionupgrade\upload 下的压缩包使用 unzipUpdate 方法进行解压

```java
public HashMap unZip(HttpServletRequest var1, HttpServletResponse var2) throws FileNotFoundException, IOException {
    boolean var3 = true;
    HashMap var4 = null;
    String var5 = (String)var1.getAttribute( s: "filename");
    File var7;
    if (var5 != null && !"".equals(var5)) {...} else {
        File var6 = null;
        var7 = new File(GCONSTUClient.getUploadSysFilePath4cluster());
        if (var7.exists()) {
            var6 = var7.listFiles()[0];
            var5 = var6.getName();
        }

        var4 = this.UpdateOperation.unzipUpdate(var6.getPath(), var5);
    }

    return var4;
}
```

此处进行了解压的操作

```java
public HashMap<String, String> unzipUpdate(String var1, String var2) {
    HashMap var3 = new HashMap();
    String var4 = "";
    if (!"".equals(var1)) {
        boolean var5 = ZipUtils.checkZip(var1);
        if (var5) {
            String var6 = GCONSTUClient.getTempSysFilePath();
            this.logger.info( o: "补丁包文件:" + var1);
            ZipUtils.deleteFile(var6,  b: false);

            try {
                ZipUtils.unZip(var1, var6);
            } catch (Exception var12) {
                var4 = "升级包处理失败，请检查升级包是否正确!";
                this.logger.info( o: "解压升级压缩包出错 : " + var12.toString());
                this.iscanupdate = false;
            }
```

而解压的目录位置则是在 ecology\versionupgrade\temp 下， 从web上直接就可以访问