CVE-2022-25237_Bonitasoft_Platform_RCE 漏洞

项目介绍

Bonitasoft 是一个业务自动化平台，可以更轻松地在业务流程中构建、部署和管理自动化应用程序； Bonita 是一个用于业务流程自动化和优化的开源和可扩展平台。

漏洞描述

在 Bonitasoft Authorization 漏洞版本，由于 API 授权过滤器中配置问题，通过精心构造的的字符串附加到 API URL，能够绕过权限认证。拥有普通用户权限的攻击者在绕过权限认证后，将恶意代码部署到服务器上，进行远程代码执行。

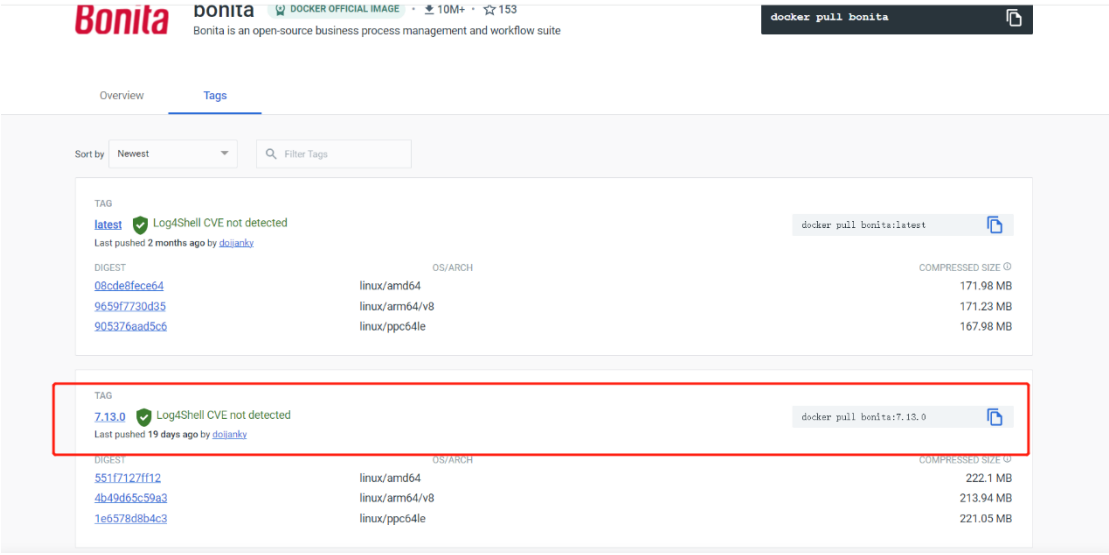
利用范围

For community（社区版）:
2022.1-u0 (7.14.0) 以下
For subscription（订阅版）:
2022.1-u0 (7.14.0) 以下
2021.2-u4 (7.13.4) 以下
2021.1-0307 (7.12.11) 以下
7.11.7 以下

漏洞分析

环境搭建

使用 docker 搭建环境，docker 镜像选择 7.13.0 版本

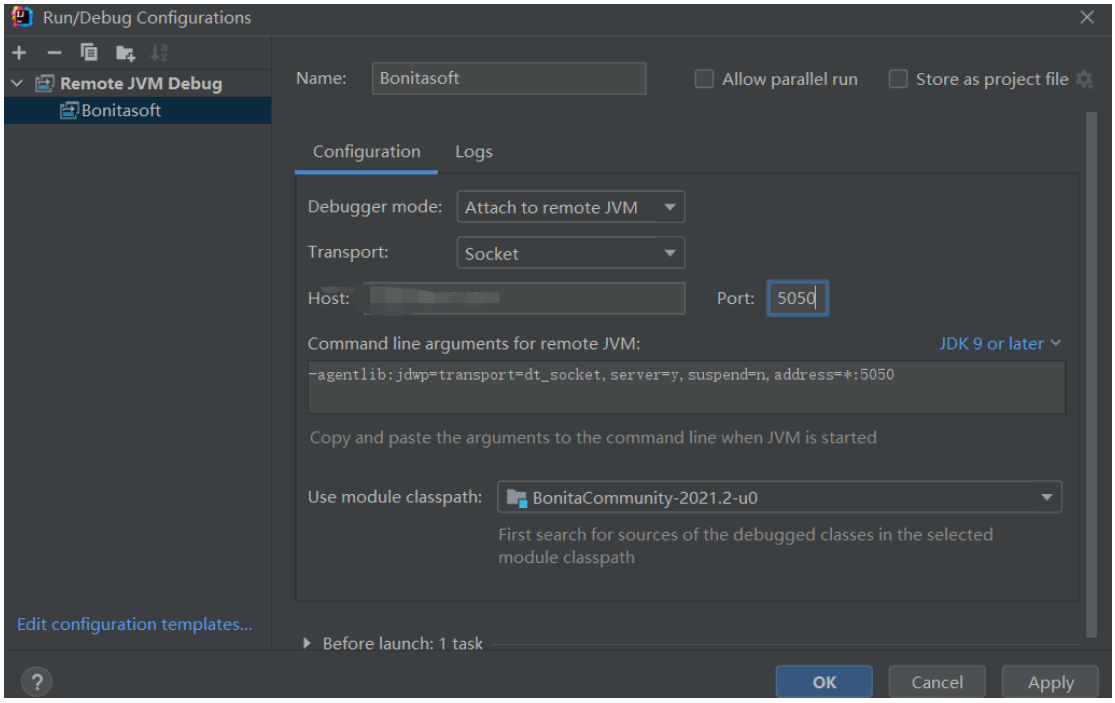


下载镜像之后，直接输入命令 docker run -d -p 8080:8080 -p 5050:5050，5050 为远程动态调试端口

在/opt/bonita 目录下载源码

```
root@ubuntu:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
 NAMES
3b7379744776        bonita:7.13.0       "/opt/files/startup.…"   About an hour ago   Up About an hour    0.0.0.0:5050->5050/tcp, 0.0.0.0:8
 compassionate_mcnulty
root@ubuntu:~# docker cp 3b7379744776:/opt/bonita /home
```

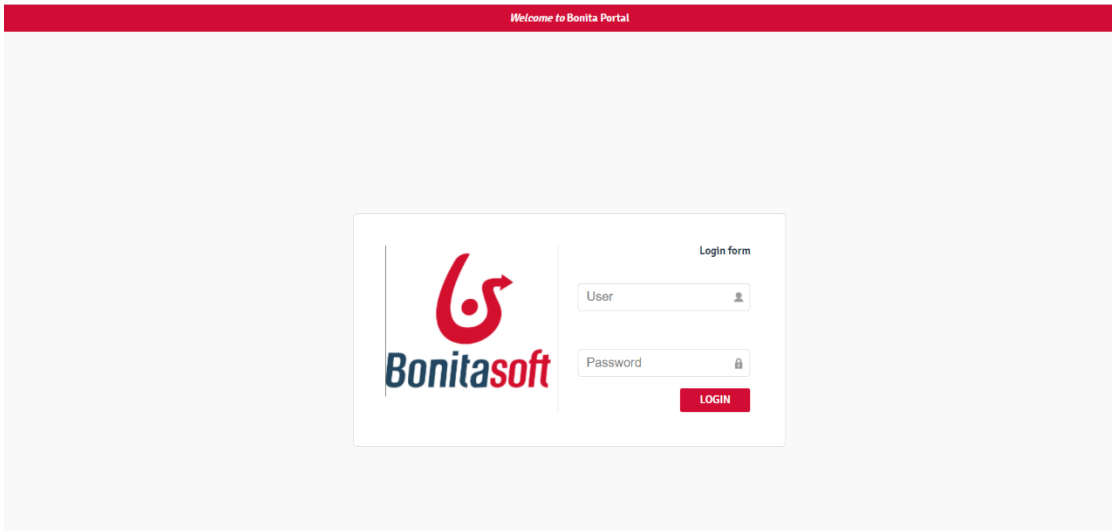
使用 idea 进行远程调试配置



在/opt/bonita/BonitaCommunity-2021.2-u0/server/bin/catalina.sh 中加入配置

```
#                   Example (all one line)
#                   CATALINA_LOGGING_CONFIG="-Djava.util.logging.config.file=$CATALINA_BASE/conf/logging.properties"
#
#   LOGGING_CONFIG  Deprecated
#                   Use CATALINA_LOGGING_CONFIG
#                   This is only used if CATALINA_LOGGING_CONFIG is not set
#                   and LOGGING_CONFIG starts with "-D..."
#
#   LOGGING_MANAGER (Optional) Override Tomcat's logging manager
#                   Example (all one line)
#                   LOGGING_MANAGER="-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager"
#
#   UMASK           (Optional) Override Tomcat's default UMASK of 0027
#
#   USE_NOHUP       (Optional) If set to the string true the start command will
#                   use nohup so that the Tomcat process will ignore any hangup
#                   signals. Default is "false" unless running on HP-UX in which
#                   case the default is "true"
# ---------------------------------------------------------------------------
JAVA_OPTS='-agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=*:5050'
# OS specific support.  $var _must_ be set to either true or false.
cygwin=false
darwin=false
os400=false
hpux=false
case "`uname`" in
CYGWIN*) cygwin=true;;
Darwin*) darwin=true;;
OS400*) os400=true;;
HP-UX*) hpux=true;;
esac

# resolve links - $0 may be a softlink
PRG="$0"

while [ -h "$PRG" ]; do
  ls=`ls -ld "$PRG"`
  link=`expr "$ls" : '.*-> \(.*\)$'`
  if expr "$link" : '/.*' > /dev/null; then
"catalina.sh" 688L, 25374C
```

随即访问 http://ip:8080，环境搭建成功



默认账号密码：install/install

代码调试

从漏洞披露参考文章看，该漏洞产生的原理是由于身份验证/权限绕过，导致没有特权的用户可以通过 api 端口上传恶意文件，从而命令执行。

Vulnerability Overview

Bonita Web 2021.2 is affected by an authentication/authorization bypass vulnerability due to an overly broad filter pattern used in the API authorization filters.

By appending a crafted string to the API URL, users with no privileges can access privileged API endpoints. This can lead to remote code execution by abusing the privileged API actions to deploy malicious code onto the server.

Affected Product

Vendor: Bonitasoft
Product: Bonita Platform
Confirmed Vulnerable Version: < 2022.1-u0
Fixed Versions:
For community:

- 2022.1-u0 (7.14.0)

For subscription:

- 2022.1-u0 (7.14.0)
- 2021.2-u4 (7.13.4)
- 2021.1-0307 (7.12.11)
- 7.11.7

Vulnerable Versions: Official Docker image

定位到参考文章所提交到的 filers
在 bonita\BonitaCommunity-2021.2-u0\server\webapps\bonita\WEB-INF\web.xml 配置文件中，自定义了 filter，并且对参数 excludePatterns 进行了赋值

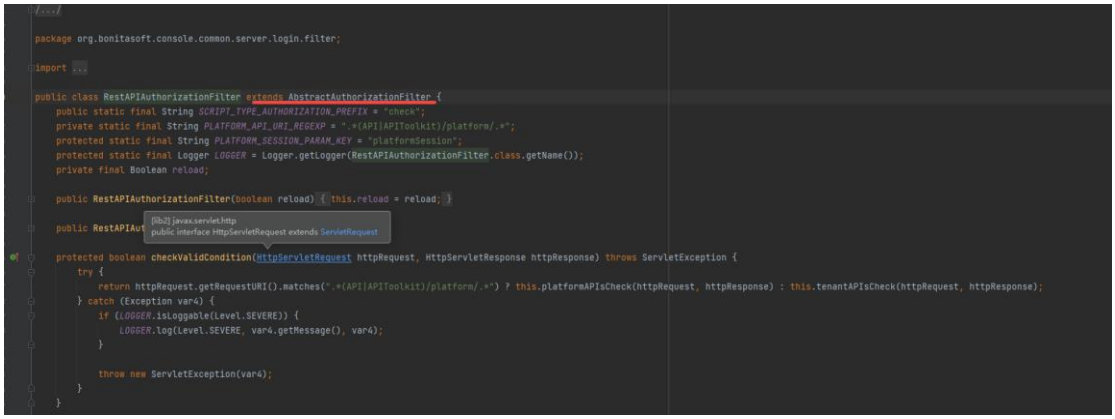
```
<!--Rest filter -->
<filter>
    <filter-name>RestAPIAuthorizationFilter</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.RestAPIAuthorizationFilter</filter-class>
    <init-param>
        <param-name>excludePatterns</param-name>
        <param-value>i18ntranslation</param-value>
    </init-param>
</filter>
<filter>
    <filter-name>RestAPIAuthorizationFilterToolkit</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.RestAPIAuthorizationFilter</filter-class>
    <init-param>
        <param-name>excludePatterns</param-name>
        <param-value>i18ntranslation</param-value>
    </init-param>
</filter>
<!-- Token Filter -->
<filter>
    <filter-name>TokenGeneratorFilter</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.TokenGeneratorFilter</filter-class>
</filter>
<!-- Token Validator Filter -->
<filter>
    <filter-name>TokenValidatorFilter</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.TokenValidatorFilter</filter-class>
    <init-param>
        <param-name>excludePatterns</param-name>
        <param-value>i18ntranslation,session</param-value>
    </init-param>
</filter>
<filter>
    <filter-name>AuthenticationFilter</filter-name>
    <filter-class>org.bonitasoft.console.common.server.login.filter.AuthenticationFilter</filter-class>
    <!--
        The AuthenticationFilter check credentials when access is requested
        However, to ensure authentication redirect and/or error handling works properly,
        we need to let a set of pages not securized :
        The excludePattern default value is the one commented out bellow, but you can uncomment it and add new patterns if you need to
    -->
    <!-- init-param>
        <param-name>excludePattern</param-name>
        <param-value>^/(bonita/)?(login.jsp$)|(apps/.+/API/)|(portal/resource/.+/API/)</param-value>
    </init-param -->
    <init-param>
        <param-name>redirectWhenUnauthorized</param-name>
```

2 个不同的过滤器类 RestAPIAuthorizationFilter、TokenValidatorFilter 中的参数 excludePattern 都被指定为“i18ntranslation”

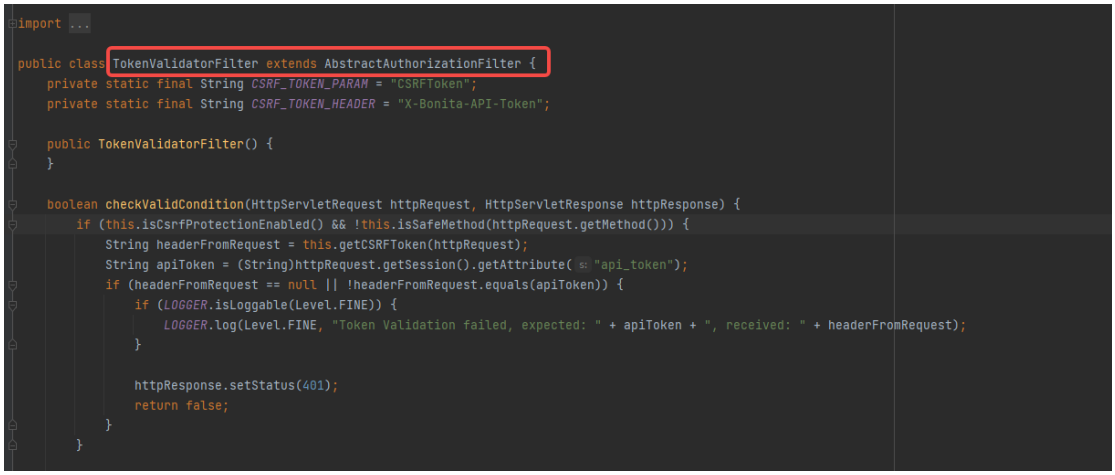
```
    <filter-mapping>
        <filter-name>TokenValidatorFilter</filter-name>
        <url-pattern>/API/*</url-pattern>
        <url-pattern>/APIToolkit/*</url-pattern>
        <!--
```

```
214     <filter-mapping>
215         <filter-name>RestAPIAuthorizationFilter</filter-name>
216         <url-pattern>/API/*</url-pattern>
217         <url-pattern>/APIToolkit/*</url-pattern>
218         <!-- see TokenValidatorFilter comment -->
```

在 web.xml 的 192 行和 215 行定义了/API/*路由下对应的 filter 为 RestAPIAuthorizationFilter、TokenValidatorFilter
而 RestAPIAuthorizationFilter 和 TokenValidatorFilter 都属于 AbstractAuthorizationFilter 子类

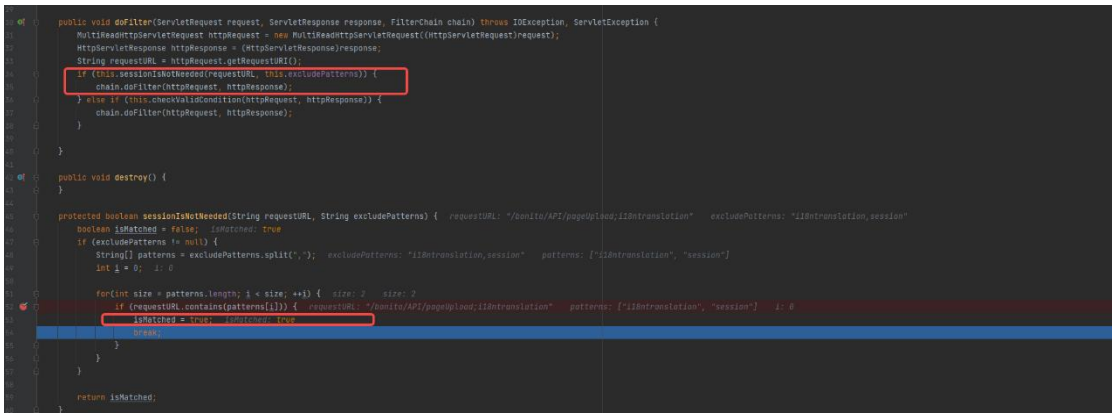


```
/.../

package org.bonitasoft.console.common.server.login.filter;

import ...

public class RestAPIAuthorizationFilter extends AbstractAuthorizationFilter {
    public static final String SCRIPT_TYPE_AUTHORIZATION_PREFIX = "check";
    private static final String PLATFORM_API_URI_REGEXP = ".*(API|APIToolkit)/platform/.*";
    protected static final String PLATFORM_SESSION_PARAM_KEY = "platformSession";
    protected static final Logger LOGGER = Logger.getLogger(RestAPIAuthorizationFilter.class.getName());
    private final Boolean reload;

    public RestAPIAuthorizationFilter(boolean reload) { this.reload = reload; }

    public RestAPIAut  [lib2] javax.servlet.http
                       public interface HttpServletRequest extends ServletRequest

    protected boolean checkValidCondition(HttpServletRequest httpRequest, HttpServletResponse httpResponse) throws ServletException {
        try {
            return httpRequest.getRequestURI().matches(".*(API|APIToolkit)/platform/.*") ? this.platformAPIsCheck(httpRequest, httpResponse) : this.tenantAPIsCheck(httpRequest, httpResponse);
        } catch (Exception var4) {
            if (LOGGER.isLoggable(Level.SEVERE)) {
                LOGGER.log(Level.SEVERE, var4.getMessage(), var4);
            }

            throw new ServletException(var4);
        }
    }
```



```
import ...

public class TokenValidatorFilter extends AbstractAuthorizationFilter {
    private static final String CSRF_TOKEN_PARAM = "CSRFToken";
    private static final String CSRF_TOKEN_HEADER = "X-Bonita-API-Token";

    public TokenValidatorFilter() {
    }

    boolean checkValidCondition(HttpServletRequest httpRequest, HttpServletResponse httpResponse) {
        if (this.isCsrfProtectionEnabled() && !this.isSafeMethod(httpRequest.getMethod())) {
            String headerFromRequest = this.getCSRFToken(httpRequest);
            String apiToken = (String)httpRequest.getSession().getAttribute( s: "api_token");
            if (headerFromRequest == null || !headerFromRequest.equals(apiToken)) {
                if (LOGGER.isLoggable(Level.FINE)) {
                    LOGGER.log(Level.FINE, "Token Validation failed, expected: " + apiToken + ", received: " + headerFromRequest);
                }

                httpResponse.setStatus(401);
                return false;
            }
        }
```

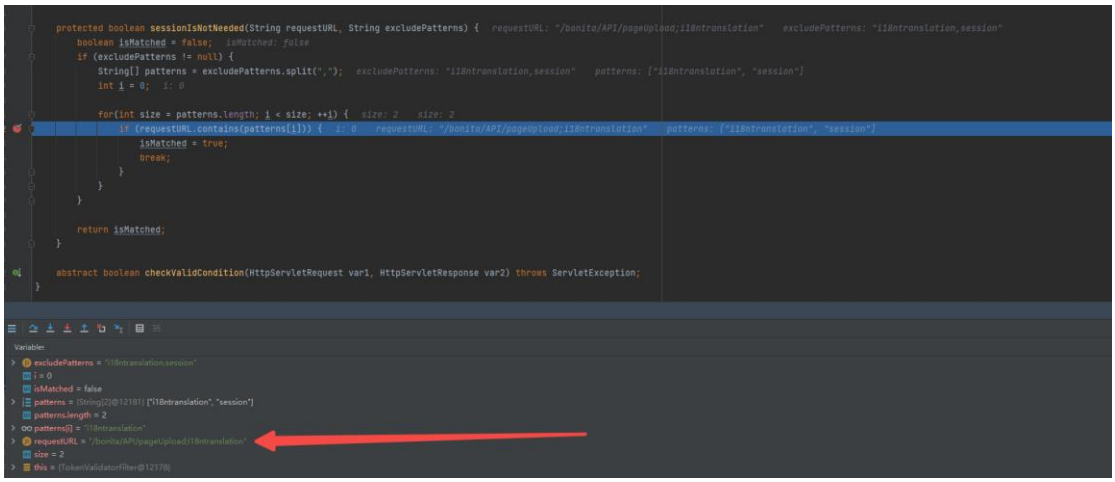
定位到 org.bonitasoft.console.common.server.login.filter.AbstractAuthorizationFilter#doFilter 函数



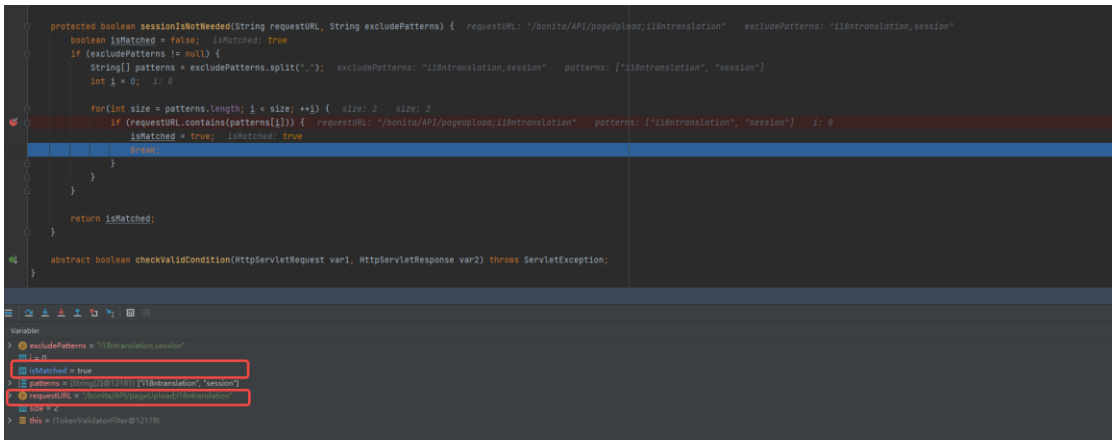
```
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {
        MultiReadHttpServletRequest httpRequest = new MultiReadHttpServletRequest((HttpServletRequest)request);
        HttpServletResponse httpResponse = (HttpServletResponse)response;
        String requestURL = httpRequest.getRequestURI();
        if (this.sessionIsNotNeeded(requestURL, this.excludePatterns)) {
            chain.doFilter(httpRequest, httpResponse);
        } else if (this.checkValidCondition(httpRequest, httpResponse)) {
            chain.doFilter(httpRequest, httpResponse);
        }

    }

    public void destroy() {
    }

    protected boolean sessionIsNotNeeded(String requestURL, String excludePatterns) {  requestURL: "/bonita/API/pageUpload;i18ntranslation"   excludePatterns: "i18ntranslation,session"
        boolean isMatched = false;  isMatched: true
        if (excludePatterns != null) {
            String[] patterns = excludePatterns.split(",");  excludePatterns: "i18ntranslation,session"   patterns: ["i18ntranslation", "session"]
            int i = 0;  i: 0

            for(int size = patterns.length; i < size; ++i) {  size: 2   size: 2
                if (requestURL.contains(patterns[i])) {  requestURL: "/bonita/API/pageUpload;i18ntranslation"   patterns: ["i18ntranslation", "session"]   i: 0
                    isMatched = true;  isMatched: true
                    break;
                }
            }
        }

        return isMatched;
    }
```

会使用“sessionIsNotNeeded”函数进行检查，如果它返回 true，它将继续应用程序流程。

而 sessionIsNotNeeded 函数检查匹配 URL 中的是否包含 excludePatterns
在 web.xml 已经对 excludePatterns 进行了赋值为“i18ntranslation”，所以当 url 中包含“i18ntranslation”时，isMatched 就会返回 true



流程就会继续，就相当于绕过了权限过滤，允许访问资源。



总结就是在 URl 包含“/i18ntranslation/../”或“;i18ntranslation”就可以绕过权限认证。
为实现远程命令执行，思路是上传恶意文件，上传接口在 web.xml 中也定义了，为/API/pageUpload

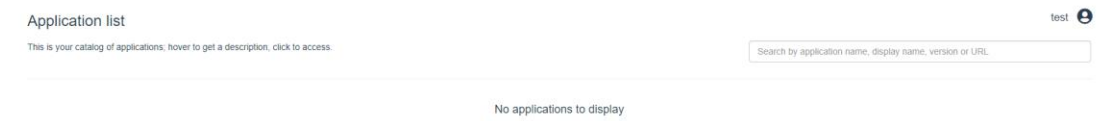
```
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>pageUploadServlet</servlet-name>
        <url-pattern>/portal/pageUpload</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>apiPageUploadServlet</servlet-name>
        <url-pattern>/API/pageUpload</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
```

找到
org.bonitasoft.console.common.server.servlet.PageUploadServlet#getPagePermissions
在文件处理过程中，需要获取的 session 存在 apisession

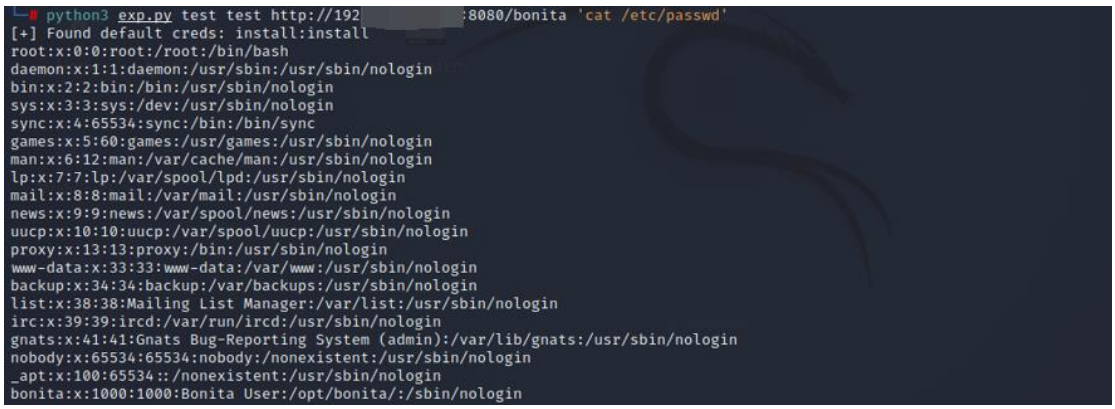
而且未登录的情况下，apisession 也没法赋值，就会导致 getPagePermissions 抛出异常。
所以，为实现远程命令执行，还需要拥有一个普通的用户。

漏洞复现

先创建一个 test/test 用户



根据以上分析原理，使用 exp，成功执行命令



修复建议

更新至安全无漏洞版本。